1997-12

Design, construction, and operation of the Naval Postgraduate School's Ultraviolet Imaging Spectrometer (NUVIS)

Hooks, Todd A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN, CONSTRUCTION, AND OPERATION OF THE NAVAL POSTGRADUATE SCHOOL'S ULTRAVIOLET IMAGING SPECTROMETER (NUVIS)

by

Todd A. Hooks

December 1997

Thesis Advisor: David D. Cleary

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1997	Master's Thesis

4. TITLE AND SUBTITLE
DESIGN, CONSTRUCTION, AND OPERATION OF THE NAVAL POSTGRADUATE SCHOOL'S ULTRAVIOLET IMAGING SPECTROMETER (NUVIS)

5. FUNDING NUMBERS

6. AUTHOR(S)
Todd A. Hooks

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER:

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER:

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Hyperspectral imaging spectrometers produce an image, comprised of the standard two-dimensional spatial scene and the corresponding spectra of each scene. Hyperspectral imaging is a relatively new and fast growing field with both commercial and military applications. Commercial applications vary from vegetation identification and mapping, surface geological identification and mapping to atmospheric composition and mapping. Military applications include target identification and classification, airborne chemical identification and mapping, and rocket plume identification.

This thesis describes the design and operation of the NPS Ultraviolet Imaging Spectrometer (NUVIS). NUVIS is a hyperspectral imaging spectrometer designed to investigate the ultraviolet region of the spectrum. NUVIS is comprised of a scanning mirror, telescope assembly using an off-axis parabolic mirror, a slit, a flat-field imaging diffraction grating, an image intensified camera assembly, and the support/controlling electric and electronic hardware and software. This is part a continuing project to build, test and use this sensor in support of military and government agencies.

14. SUBJECT TERMS	15. NUMBER OF PAGES:
Hyperspectral Imaging, Ultraviolet, Imaging Spectrometer, NUVIS, Support to Military Operations, Support to Government Agencies	96
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)

Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

DESIGN, CONSTRUCTION AND OPERATION OF THE NAVAL POSTGRADUATE SCHOOL'S ULTRAVIOLET IMAGING SPECTROMETER

Todd A. Hooks
Lieutenant, United States Navy
B.S., The Virginia Military Institute, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL

ABSTRACT

Hyperspectral imaging spectrometers produce an image comprised of the standard two-dimensional spatial scene and the corresponding spectra of each scene. Hyperspectral imaging is a relatively new and fast growing field with both commercial and military applications. Commercial applications vary from vegetation identification and mapping, surface geological identification and mapping to atmospheric composition and mapping. Military applications include target identification and classification, airborne chemical identification and mapping, and rocket plume identification.

This thesis describes the design and operation of the NPS Ultraviolet Imaging Spectrometer (NUVIS). NUVIS is a hyperspectral imaging spectrometer designed to investigate the ultraviolet region of the spectrum. NUVIS is comprised of a scanning mirror, telescope assembly using an off-axis parabolic mirror, a slit, a flat field imaging diffraction grating, an image intensified camera assembly, and the support/controlling electric and electronic hardware and software. This is part of a continuing project to build, test and use this sensor in support of military and government agencies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

The author would like to acknowledge the financial support of HYMSMO, for making NUVIS possible.

I. INTRODUCTION

Imaging spectrometry is a new and rapidly growing form of spectrometry. Unlike traditional spectrometry, where only spectral data is obtained, imaging spectrometry produces a two dimension spacial image with the corresponding spectral data. Numerous multi-spectral and several hyper-spectral instruments have been constructed to investigate the visible through infrared regions of the spectrum, but the ultraviolet (UV) region of the spectrum has all but been ignored. Naval Postgraduate School (NPS) UltraViolet Imaging Spectrometer (NUVIS), a hyperspectral imaging spectrometer, was designed to investigate and map the emission, absorption, and reflection of solar UV radiation by the earth's surface, gas clouds, and military targets. The ability to combine both the spectral and spacial data allows large areas to be imaged and analyzed. Once a characteristic signature has been determined, data analysis allows specific objects to be identified and located. One of NUVIS' first applications will be to analysis the SO_2 gas and ash emissions from volcanos. Military applications include battle field target identification and location through the use of Unmanned Aerial Vehicles.

NUVIS started in 1990 with an instrument called MUSTANG (Middle Ultraviolet SpecTrograph for Analysis of Nitrogen Gases), designed to analyze the earth's ionosphere. MUSTANG utilized an Ebert-Fastie spectrometer with a diode detection array. For more information see Walden (1991). Johnson (1996) attempted to modify MUSTANG to produce an imaging spectrometer. The modified instrument was named Dual Use UltraViolet Imaging Spectrometer (DUUVIS). Johnson finished DUUVIS, but did not have time to conduct tests do determine the quality of the images. During January-February 1997,

tests were conducted on DUUVIS by MacMannis (1997) and myself. We discovered that DUUVIS could not be modified to produce acceptable images, due to optical limitations. At about the same time, funding was obtained from Hyperspectral Measurement and Signals Intelligence Support for Military Operations (HYMSMO) to build a new instrument from scratch.

From this point the design of NUVIS branched into two areas. I became responsible for all electronics, the scanning mirror, baffles, filters, software, and the position of these components. Andy MacMannis was responsible for the off-axis parabolic telescope mirror, slit, and the diffraction grating. The telescope mirror, slit, and diffraction grating will be covered briefly, but for detailed information see MacMannis (1997).

The telescope mirror was a mirror we already had on hand. It therefore became the starting point for the entire design layout, size, and optical component selection. My work really began from here.

II. OPTICAL COMPONENTS

Figure 1 shows a line drawing of the final design of the imaging spectrometer NUVIS. The optical components, marked 1 through 11, are identified in the figure caption and described in detail below. All of these components are securely attached to the base plate.

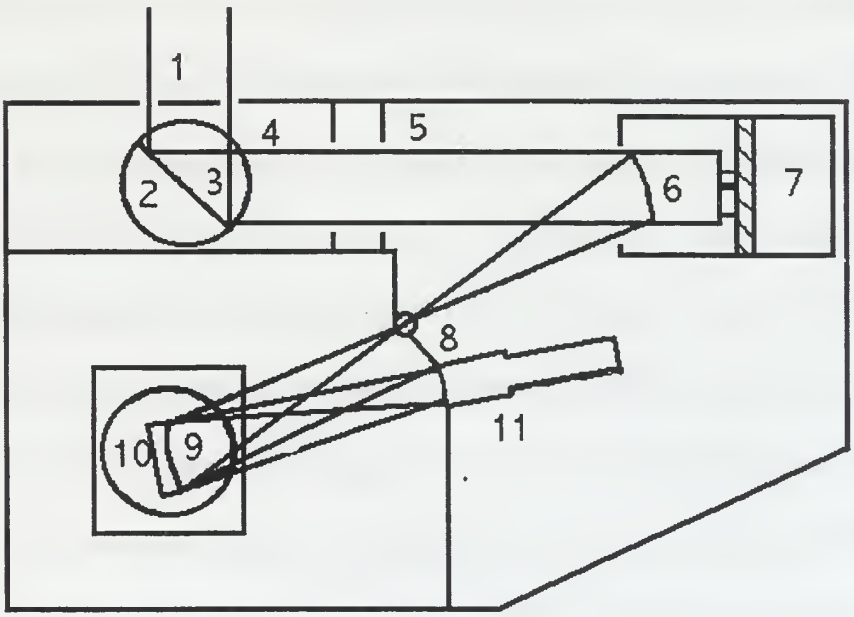


Figure 1: Optical Path in NUVIS; (1) filter window, (2) scanning mirror housing, (3) scanning mirror, (4) baffle, (5) baffle, (6) telescope mirror, (7) telescope mirror housing, (8) slit, (9) diffraction grating, (10) diffraction grating holder, (11) image intensified camera. After MacMannis (1997).

A. TELESCOPE MIRROR

The telescope mirror was an off-axis parabolic mirror manufactured by Space Optics Research Labs (SORL), sales order number SN4338. The mirror was 50.8 mm in diameter,

had a focal length of 152.4 mm with an off axis distance of 63.5 mm. The surface was coated with aluminum, and was overcoated with MgF_2 providing a reflectivity greater than 95% for wavelengths above 280 nm. The scratch/dig was 60/40, and the surface accuracy was $\lambda/15$ at 632.8 nm. SORL also provided the telescope mirror housing. With the mirror mounted in the housing, the optical axis was 3.5 inches above the top of the base plate. Figure 1 shows the telescope mirror along with the other optical components.

B. SLIT

The spectrometer slit dimensions were determined by the focal length of the diffraction grating and the size of the camera's CCD pixels as projected on the image intensifier input window. The width of the slit was arbitrarily decided to be three CCD pixels, which was approximately 60 μm. At this point in NUSIS's design, there was no way to determine whether a 60 μm slit would produce the desired image on the intensifier photocathode or simply limit the intensity of the light. A quick look at the geometry showed that the width of the slit would be the limiting factor in determining the sizes of the baffles, scanning mirror, and entrance window. Therefore, it was decided to make the slit 90 μm wide, or 4.5 pixels on the image intensifier.

Next the height of the slit had to be determined. The diffraction grating provided a flat field focus in the 300 - 400 nm spectral range over a distance of 25 mm. Since the height of the camera's CCD as projected on the photocathode was 17.7 mm, anything outside this would be lost. With a maximum usable height of 17.7 mm we chose the slit height to be 15 mm, which allows for slight misalignments.

A search found no commercially available slits that met both the width and the height

required. It was decided to fabricate the slit using razor blades. See MacMannis (1997) for further details on the slit.

A holder had to be constructed for the slit. The original holder was a post holder with the slit at the top. As construction continued, it was found that this type of holder did not allow for a light tight enclosure. A new holder was designed that would fasten to both the image intensifier housing and one of the internal light tight walls. The holder had two purposes: 1)It had to hold the slit, and 2)It had to provide a light tight seal around both the slit and the image intensifier. An added benefit was that by fixing the relative position of the slit and the image intensifier a degree of freedom was removed, therefore making optical alignment simpler.

C. INSTANTANEOUS FIELDS OF VIEW

Using the dimensions of the slit and the focal point of the telescope mirror the Instantaneous Fields Of View (IFOV) were be calculated. The Equations:

$$\theta = \text{Slit Height} / \text{Telescope Mirror Focal Length}$$
$$\phi = \text{Slit Width} / \text{Telescope Mirror Focal Length}$$

were used to calculation the IFOVs. Where ϕ is the horizontal IFOV and θ is the vertical IFOV. From these equations I determined ϕ to be 0.59 milliradians and θ to be 98.5 milliradians. The IFOVs are then used to determine the size of the baffles, scanning mirror, and entrance window.

D. BAFFLES

In order to reduce stray light, two baffles were placed between the scanning mirror and the telescope mirror. The baffles were located 317.5 mm and 381 mm from the rear edge

of the base plate and centered along the optical axis, as shown in Figure 1. The size of the baffles are calculated by using the IFOVs. Since the baffles were not intended to be the limiting aperture, 10 mm was added to each dimension. For the baffle placed at 317.5 mm, the horizontal and vertical dimensions are 61 mm and 76 mm respectively. For the baffle set at 381 mm, the horizontal and vertical dimensions were 61 mm and 83 mm respectively. The edge facing the telescope mirror was tapered at 30 degrees. The edge facing the scanning mirror had no tapper, which created a knife edge to reduce reflection. Figure 2, is the drawing for the baffle nearest the telescope.

E. SCANNING MIRROR ASSEMBLY

The scanning mirror assembly, consists of a number of separate components: (1)the scanning mirror, (2)the scanning mirror housing, (3)the stepping motor, and (4)the absolute encoder. The stepping motor and absolute encoder are covered in Chapter 3.

1. Scanning Mirror

The size of the scanning mirror was calculated using a process that was similar to that used for the baffles, but unlike the baffle calculations, the angle of rotation of the scanning mirror had to be taken into account.

The zero angle for the scanning mirror was at 45 degrees from the telescope mirror optical axis, seen in Figure 1. The scanning arc was 10 degrees, 5 degrees to either side of the zero angle, that provided a total Field Of View (FOV) of 20 degrees.

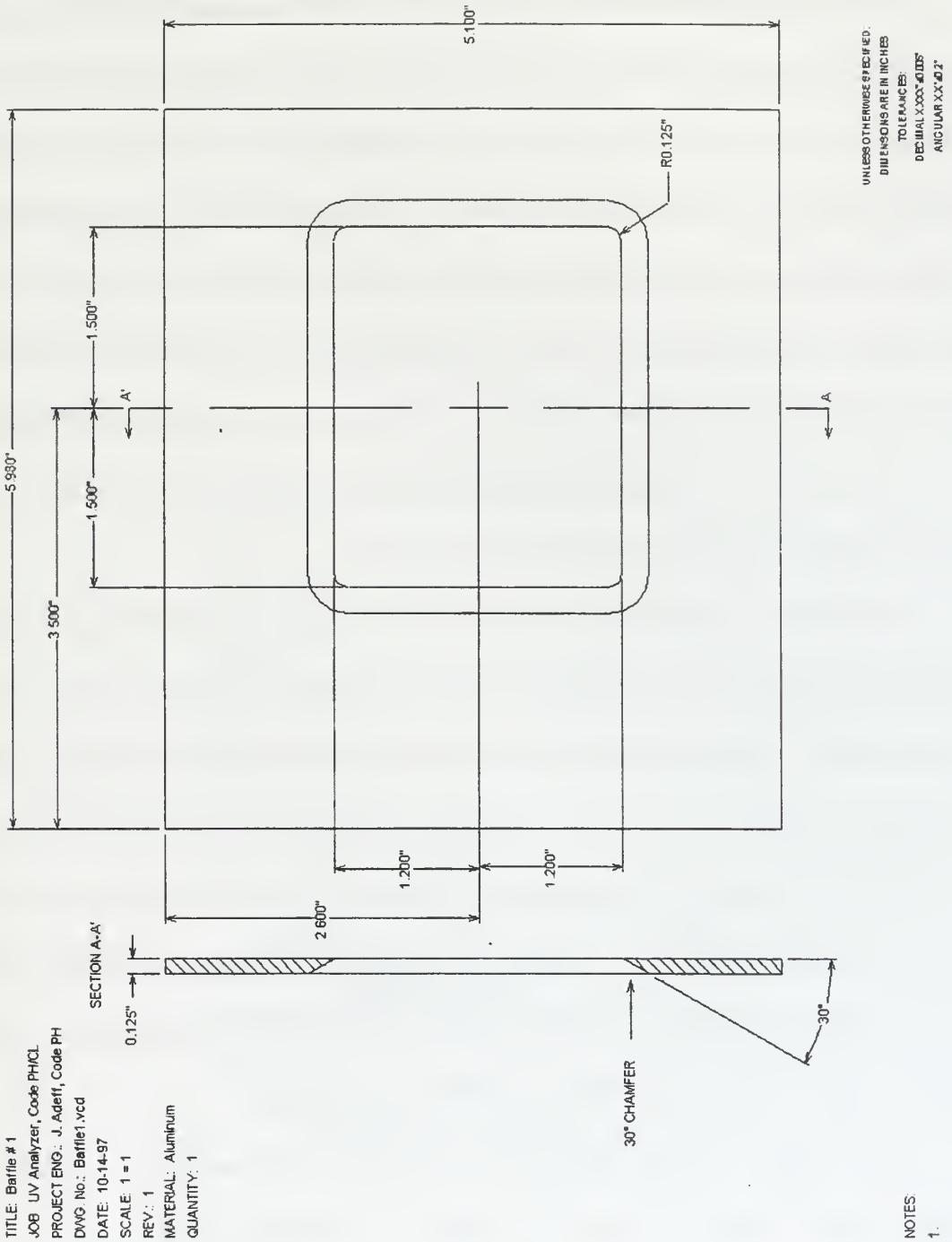


Figure 2: Baffle Drawing.

The scanning mirror had to be large enough to fill the entire IFOV when offset 50 degrees from the telescope optical axis. I decided that the mirror width would be determined by the longest optical path. By this I mean, that the optical path from the slit to the edge of the scanning mirror farthest from the telescope mirror would be used to determine the scanning mirror size to avoid vignetting. The reason for this was that if the actual optical paths had been used, the optical center of the scanning mirror would not have corresponded to the geometrical center and the scanning axis. The end result was that the mirror was wider than actually needed. The benefits of a smaller scanning mirror were more than offset by having the optical, geometric, and scanning axis coincide.

Given these considerations the minimum mirror dimensions were 80 mm wide by 84 mm high. To ensure that the mirror does not cause vignetting and to allow clearance for the mirror brackets, 10 mm was added to the width and 6 mm was added to the height, requiring a mirror that was 90 mm by 90 mm. The actual projection of the slit at the position of the scanning mirror was an ellipse, therefore the corners were cut at a 45 degree angle, 5 mm from the corner along each side. In addition, to help prevent chipping and to make handling easier, a 1 mm by 1mm bevel was placed around the entire finished side edge of the finished side of the mirror. Figure 3 is a drawing of the scanning mirror.

The company that manufactured the scanning mirror was P.A. Clausing. The scanning mirror substrate was made of Zerodeur to minimize thermal expansion. The surface was coated with aluminum and overcoated with a proprietary UV coating called UV280. The UV280 protects the surface while still allowing greater than 95 percent reflectance for wavelength above 280 nm. UV280 was choosen over MgF_2 based on the

recommendation of the mirror manufacture (Clausing, 1997). UV280 was more durable and has a higher resistance to moisture than MgF_2, making it the better choice. Our requirement for the mirror flatness was to have a deviation of no more than $\lambda/10$ at 632.8 nm across the diagonal, which required a minimum mirror thickness of 12 mm. The finished mirror exceeded the requirements with a flatness of $\lambda/20$ at 488 nm across the diagonal and a scratch/dig of 20/10. This makes the telescope mirror the limiting reflective surface in the optical path.

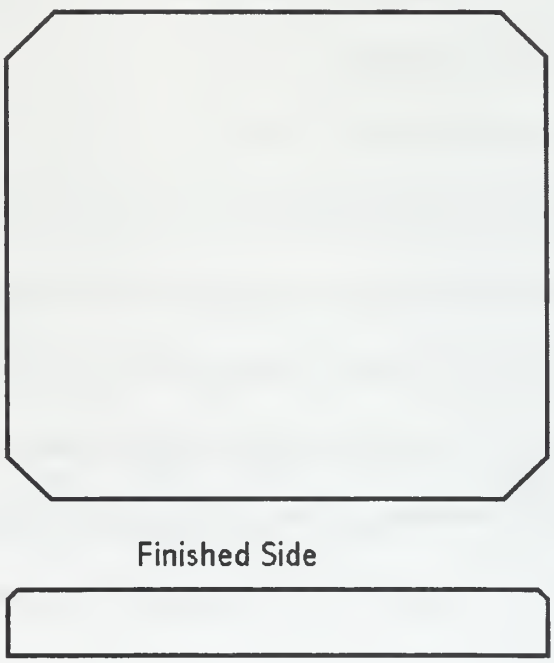


Figure 3: Scanning Mirror.

2. Scanning Mirror Housing

The scanning mirror housing, seen in Figure 4, was a 127 mm diameter aluminum tube with wall thickness of 4.8 mm. The height of the housing was 152.4 mm, with a 4.8 mm plate filling the top and another 4.8 mm plate 28.6 mm from the bottom, allowing room to mount the absolute encoder. Pressed into the center of each of the plates was a bearing, that has an internal diameter of 6.4 mm, for the scanning mirror shaft. There are two sections cut out of the side of the scanning mirror housing tube, each section was cut such that it produces a square aperture 90 mm by 90 mm. The two openings were centered on the optical axis, and were 90 degrees apart.

A scanning mirror holder, seen in Figure 5, was made of 19 mm honey comb aluminum. At the top and bottom of the holder was a 6.4 mm steel shaft that slides through the upper and lower bearings where the stepping motor and absolute encoder attached. The holder was designed such that when the scanning mirror was in the holder, its finished surface bisects the center of the scanning shaft. Along the sides of the holder was a 6 mm removable aluminum bar that over-lapped the scanning mirror by approximately 2 mm. The holder design allows the scanning mirror unlimited rotation while preventing any torque from being placed on the mirror.

Figure 4: Scanning Mirror Assembly.

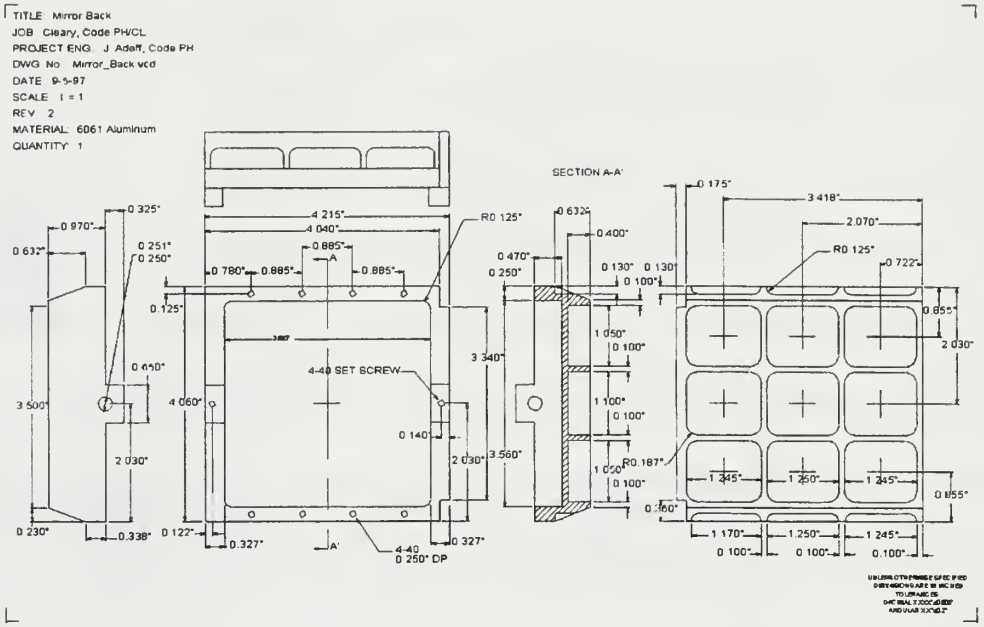


Figure 5: Scanning Mirror Holder.

F. ENTRANCE WINDOW AND FILTER

1. Entrance Window

The entrance window, shown in Figure 6, had the tightest dimensions in the optical path. The dimensions were calculated in the same manner as the baffles and scanning mirror size, except that the optical path lengths for the center and edges were calculated in detail. The result was a window with the optical axis offset from the geometrical center. The minimum height of the window was calculated to be 88 mm. The minimum width was determined to be 68 mm, 40 mm from the optical axis toward the telescope mirror and 28 mm from the optical axis away from the telescope mirror. Five millimeters were added to both the height and the width. The final dimensions were 93 mm by 73 mm (43mm and 30 mm with respect to the optical axis). Immediately outside the window was a 3.2 mm by 3.2 mm lip, designed to hold the filter.

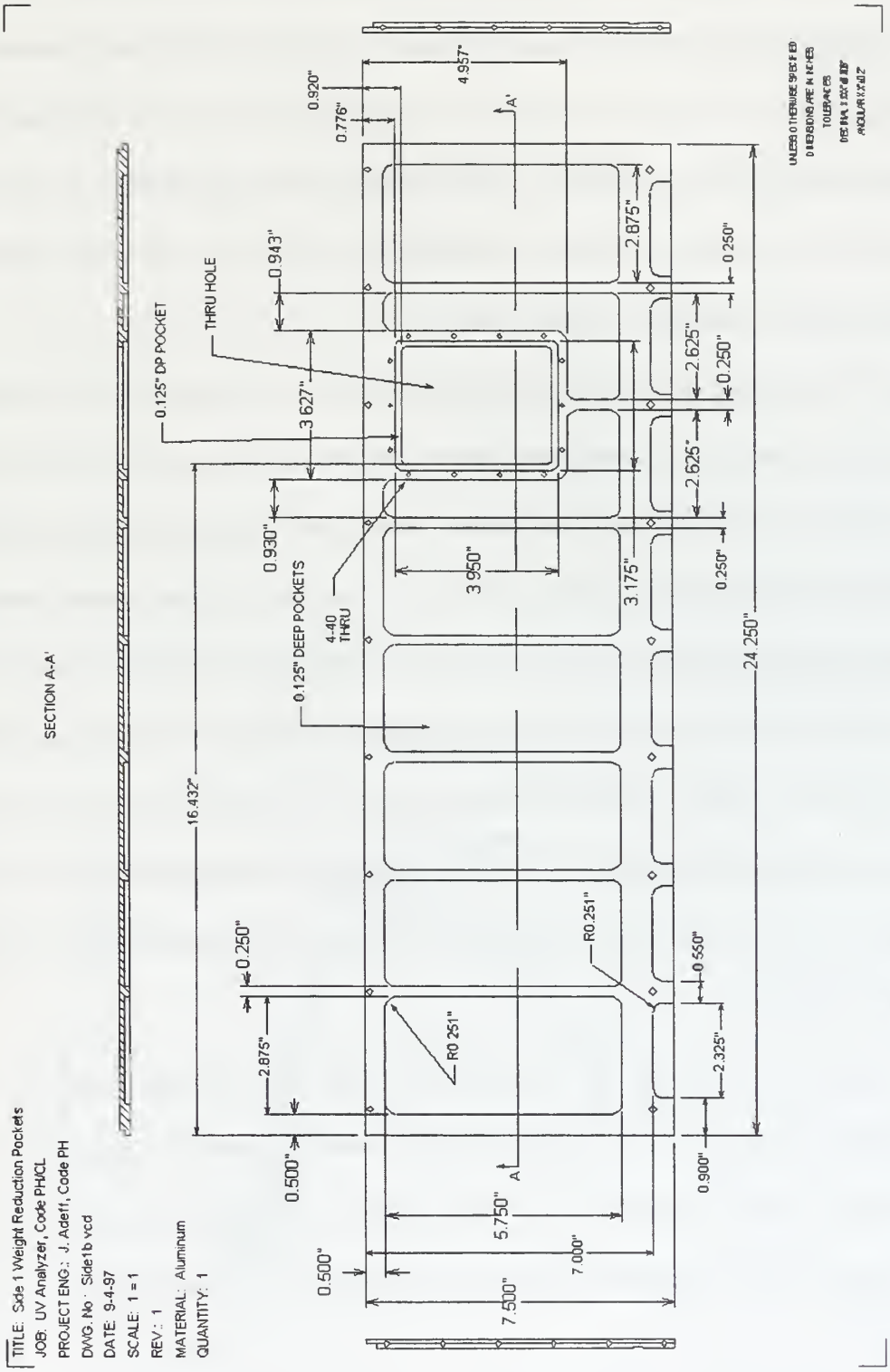


Figure 6: Entrance Window Drawing.

2. Filter

During the optical design, a single MicroChannel Plate (MCP) image intensifier was used to observe the spectrum of a mercury lamp. While using this image intensifier, it became apparent that its sensitivity to visible light would present difficulties if any stray light were present. An optical filter needed to be added, in addition to making NUVIS light tight and using baffles to reduce internal reflections.

The placement of the filter required some careful consideration. The filters with the best optical properties were interference filters. Interference filters require that incident light be collimated and be normal to the surface. This meant that any interference filter used would have to be mounted within the telescope. It would have been inconvenient to have the filter mounted internally, making it nearly impossible to change the filter while in the field. Despite this inconvenience, I researched the availability of both bandpass and short pass interference filters. I found several filters that met the optical requirements, but none that were manufactured in the sizes needed. The only way to get interference filters large enough was to have them custom made, but the cost was not worth the benefits. This forced a change in tactics.

I decided to place the filter outside the telescope, and if possible use the filter as the input window. This had two advantages: (1)it eliminated the need for a quartz window, and (2)it made it easy to change the filter. The down side was finding an absorption filter that met both the optical requirements and the even larger size requirements of the entrance window.

My search failed to find bandpass or shortpass filters that had the optical properties

needed. There were several filters that came close to what was required and were available in custom sizes, all for a reasonable cost. I decided to simulate the spectrum of the output of the image intensifier using these filters. To do this I needed the solar irradiance at the earth's surface, the transmittance curves of the filters, the reflectivity curves of the mirrors, and the sensitivity curve of the image intensifier.

The solar irradiance data at the surface, seen in Figure 7, was obtained using Lowtran 7. The two filters that appeared to be the most promising were Schott glass UG-5 and UG-11 filters. After contacting Schott directly, I obtained a data printout of the transmittance for both filters, seen in Figures 8 and 9. The mirror reflectance was greater than 95 percent for wavelengths above 280 nm for each mirror. I therefore used a constant reflectance of 95 percent. The image intensifier supplier had provided the sensitivity data with the intensifier, seen in Figure 10 (Ziemer & Associates, Inc., 1997).

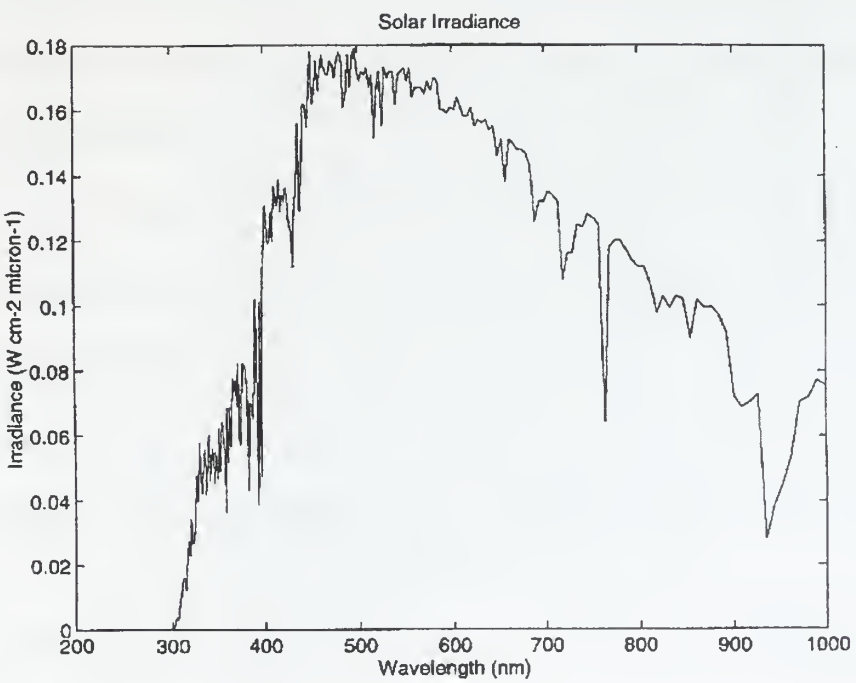


Figure 7: Solar Irradiance at Earth's Surface. From Lotran 7.

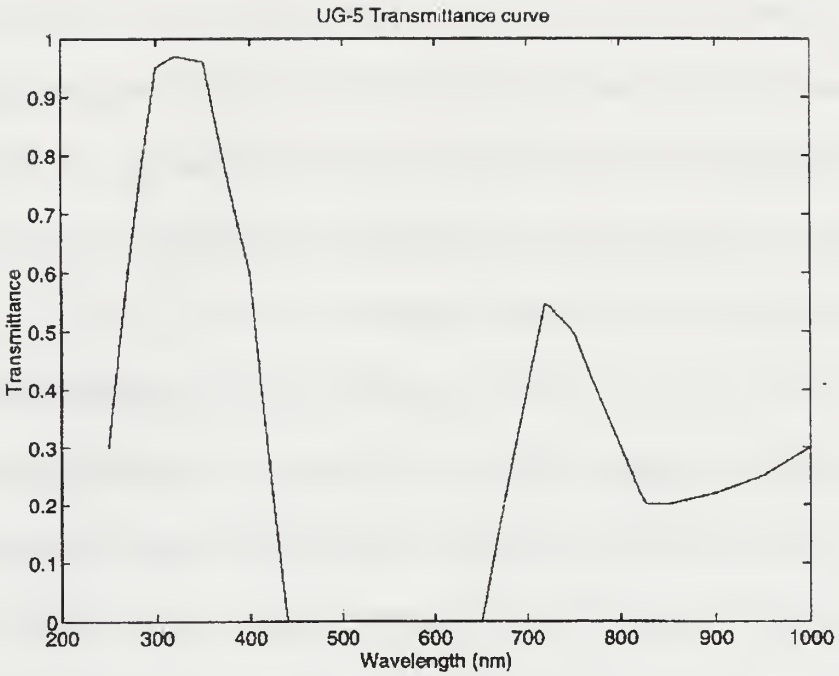


Figure 8: Transmission Curve for UG-5 Filter. From Schott.

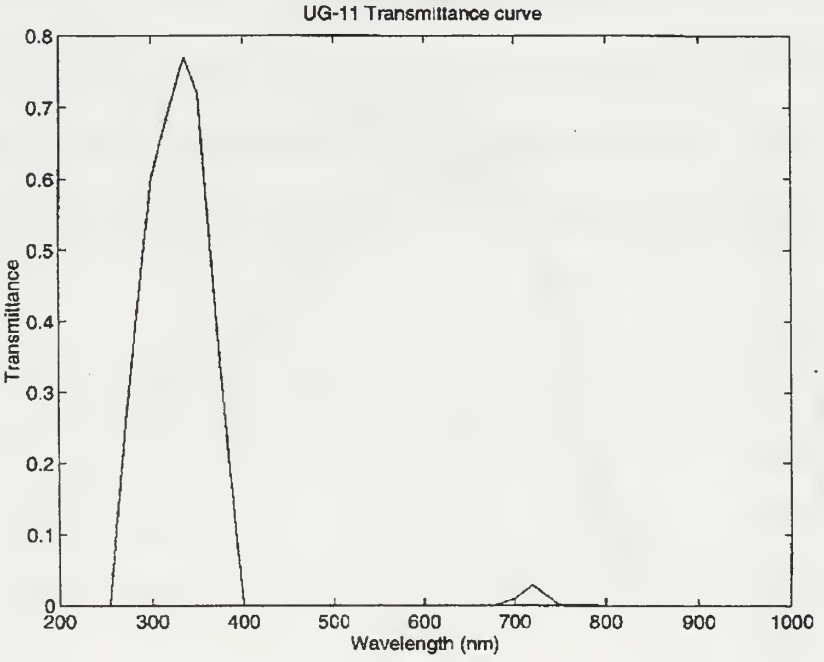


Figure 9: Transmission Curve for UG-11 Filter. From Schott.

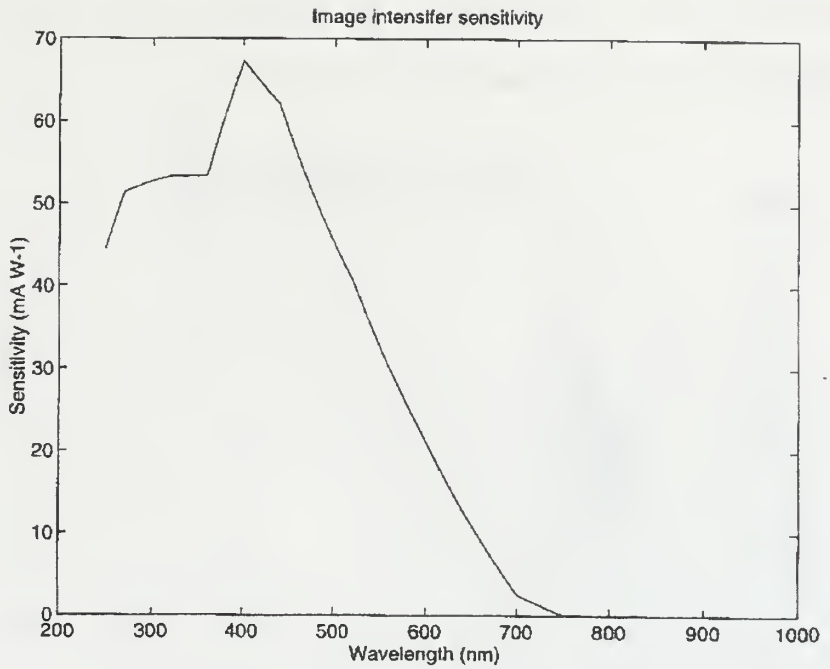


Figure 10: Image Intensifier Sensitivity. After Ziemer & Associates (1997).

With all the required data, I wrote a simple program that would read the data from text files, interpolate the data elements so that each data matrix would have the same number of elements, and then multiply all the matrixes together. The software that I used was MATLAB by MathWorks.

The results of my simulations proved surprising. Both the UG-5 and UG-11 turned out to be much better than had been originally thought. The image intensifier sensitivity with the UG-5 filter, seen in Figure 11, provided a sharp cutoff at 300 - 440 nm with only a slight sensitivity appearing at 650 - 750 nm. The image intensifier sensitivity with UG-11 filter, seen in Figure 12, had an even sharper cutoffs at 300 - 400nm, but the sensitivity was only half of that found when using the UG-5. I decided to sacrifice the out-of-band contamination, due to a wider bandwidth of the UG-5, for the higher sensitivity. Despite my

decision to use the UG-5, I felt both filters were worth purchasing, in case I experienced difficulties with the extended bandpass of the UG-5 filter.

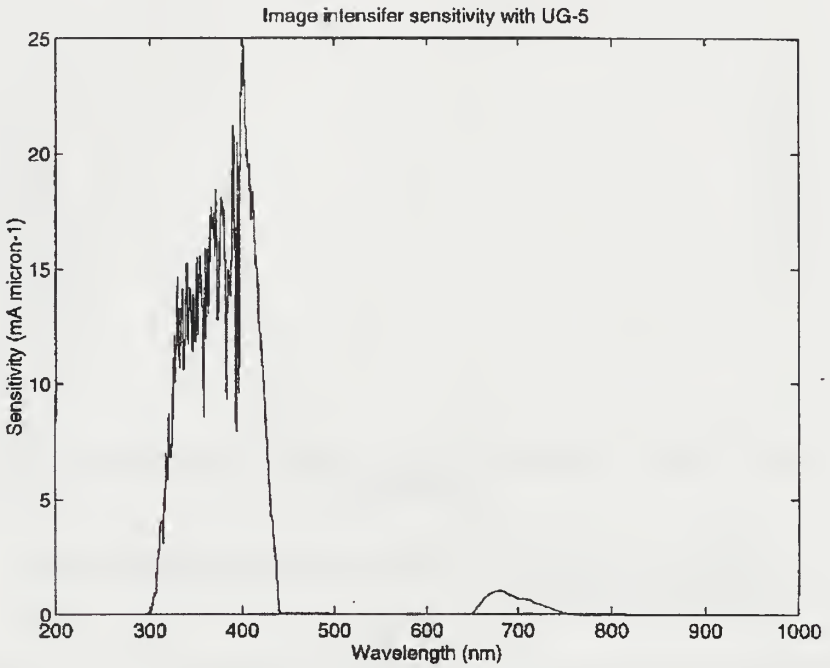


Figure 11: Sensitivity with UG-5 Filter.

The filter's physical dimensions were determined by adding the window dimensions to lip size. The result was a filter 100 mm by 80 mm, that served as a band pass filter and as a window. It had to be sufficiently thick to prevent easy breakage, therefore the thickest variant of theses filters, 3 mm, was used. Two additional physical requirements were add. The first was that all edges be slightly rounded to prevent chipping, and that the corners have a 3.2 mm curvature. The 3.2 mm curvature was to help prevent chipping, but it was primarily added to make the machining of the filter holder easier.

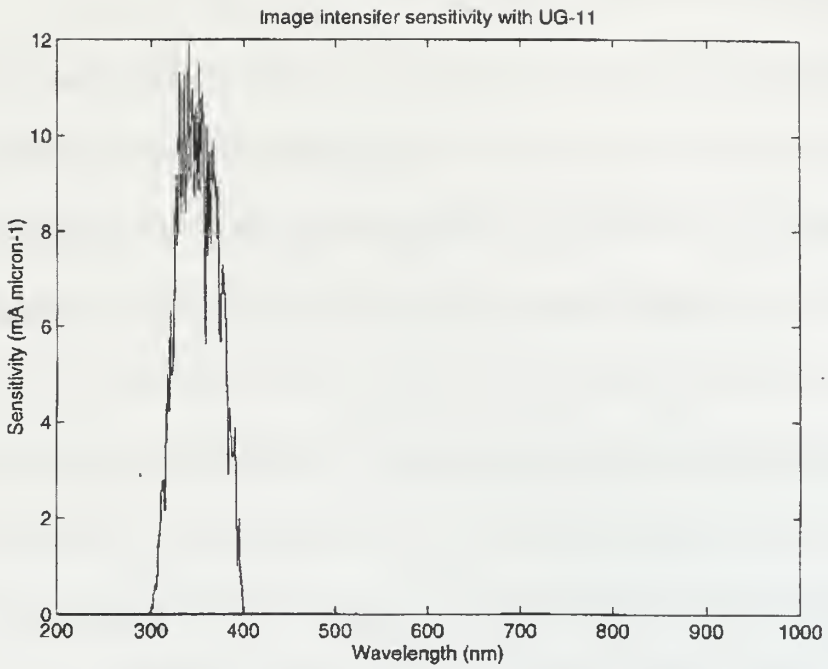


Figure 12: Sensitivity with UG-11 Filter.

Figure 13, shows the entrance window with the UG-5 filter, the missing screws on the top and bottom of the filter holder are used to hold the filter cover.

Figure 13: Entrance Window.

G. DIFFRACTION GRATING

The diffraction grating was the single most difficult optical component to obtain. This had to due with two factors: (1)the wavelengths of interest and, (2)the grating needed to have a flat field focus. MacMannis (1997) identified a supplier of a grating that would suit our needs. It was a flat field imaging spherical diffraction grating from Instruments S. A., Inc (ISA).

As discussed above, the grating provide a flat field focus in the wavelength range of 300 - 400 nm over a length of 25 mm. It had a ruling density of 1200 grooves/mm. The grating was manufactured using a Zerodeur substrate, coated with aluminum and overcoated with a UV protective coating of MgF_2 to protect the aluminum. The effective reflectivity over this bandwidth was greater than 95 percent. The grating was mounted in an Oriel holder that allows rotation of the grating about its center and about an axis passing through the center. For further details on the diffraction grating and its holder, see MacMannis (1997).

H. IMAGE INTENSIFIED CAMERA

1. Requirement for an Image Intensifier

The earth's atmosphere is relatively transparent to visible and infrared (IR) radiation but it is an effective absorber of UV radiation. Only a small fraction of UV radiation incident on the earth's atmosphere is transmitted to the surface. Therefore in order to observe the UV spectrum at or near the earth's surface, a detector that was extremely sensitive to UV light was needed. It was also important that the detector be fast, which required a detector that did not require long integration times. The best detector for this application was an image intensifier.

2. Image Intensifier Optical Parameters

The image intensifier used was a dual MCP image intensifier manufactured by Delft Electronische Producten (DEP), a Netherlands-based company. The image intensifier retains positional information and relative intensity, therefore, our spectral and spatial input was related to the output. The intensifier serves two purposes in our application: (1)it provides a usable photon-to-photon gain of about 10^5 and, (2)it produces a shift between the input and the output wavelengths, shifting the UV light to the visible spectrum to match the camera's peak sensitivity.

At the image intensifier input was the photocathode, made of modified S-20 semiconductor material. The standard S-20 semiconductor material has been modified to shift the peak sensitivity toward the UV and minimize the sensitivity in the visible and infrared regions. The photocathode was sensitive to light between 200 - 700 nm, as seen in Figure 10.

At the image intensifier output was a P-20AF phosphor screen. The screen produces output light that peaks between 500 - 630 nm, as shown in Figure 14, corresponding to the camera's peak sensitivity of 500 - 650 nm, as shown in Figure 15.

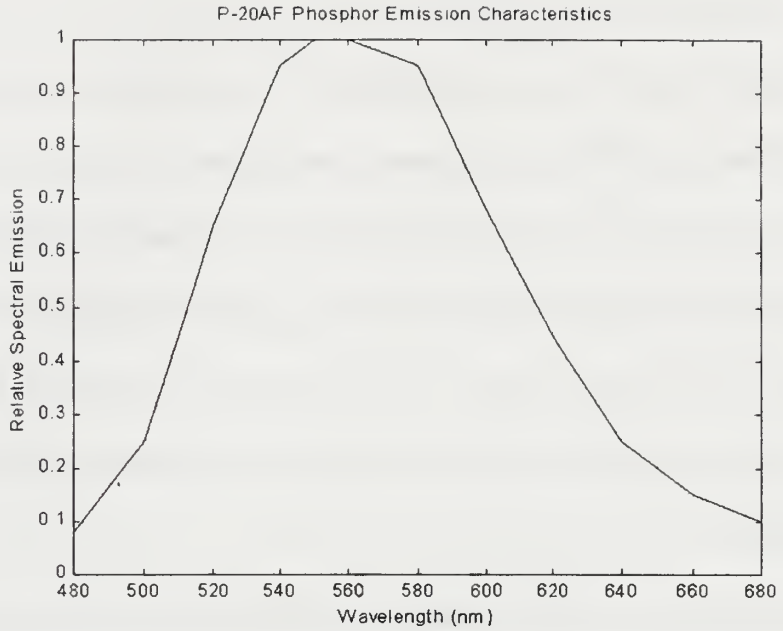


Figure 14: P-20AF Phosphor Emission Curve. From DEP (1994).

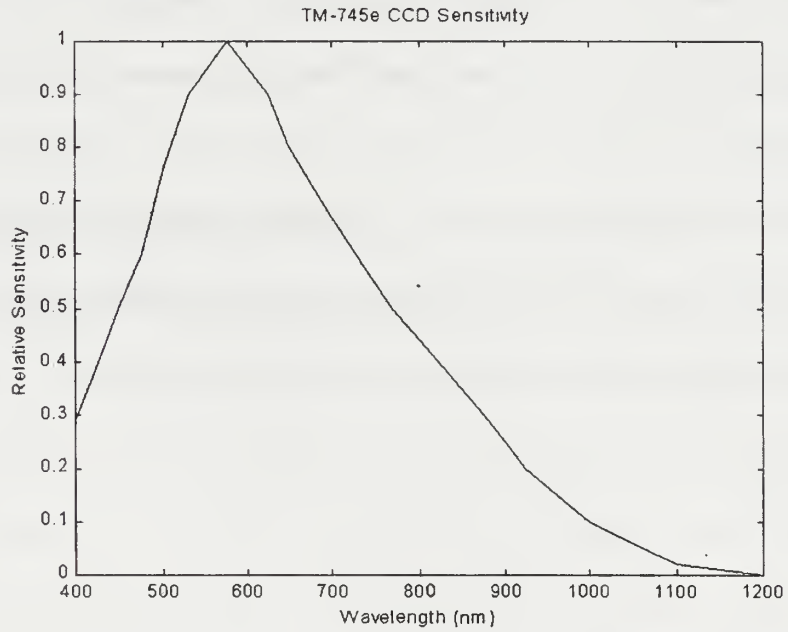


Figure 15: TM-745e CCD Sensitivity. After Pulnix (1995).

3. Fiber Optic Coupling

The image intensifier input and output windows were 25 mm in diameter, where as the camera's CCD was a rectangle 10 mm by 9.3 mm. Some form of coupling was needed between the two components. There are two choices for coupling optical components such as this, lens coupling and fiber optic coupling.

Fiber optic coupling was chosen over lens coupling because it was physically smaller, more efficient, and more durable. The fiber optic bundle or tapper was 25 mm in diameter on the intensifier side and 13.7 mm in diameter on the camera side. I decided that the camera CCD would be completely enclose withing the circle of the fiber optic tapper, i.e., the rectangle was within the circle. This allows for full utilization of the CCD while sacrificing some of area of the image intensifier. This had the effect of reducing the bandwidth by about 15 nm for a total bandwidth of 85 nm.

4. Optical Parameters of the CCD Camera

The CCD Camera used was a Pulnix TM-745e video camera with a Sony 2/3 inch CCD having 768 by 494 pixels. As mention above, the CCD measures 10 mm by 9.3 mm with pixels that were 11 μm by 13 μm and the peak sensitivity was between 500 nm and 650 nm, as shown in Figure 15.

In order to calculate the slit width I needed the pixel size as projected on the input window of the image intensifier. This was done by starting with the diagonal of the CCD, which was 13.7 mm. The diagonal was then dividing by the diameter of the image intensifier, 25 mm. The result was then multiplied by the pixel dimensions. The resultant projected pixel on the photocathode had the dimensions of 20.1 μm by 23.8 μm.

I. ENCLOSURE

With all of the optical components and their holders identified, the construction of the enclosure could begin. The first thing that was done was to machine two joining sides of the base plate, making a square corner. In this corner the telescope mirror was mounted. Using the telescope mirror, the location of each component was determined. It was my job to place the individual components, determine the dimensions of each component that had to be made, and draw sketches of these components. I didn't have the time to convert these drawings to blueprints that could be handed to a machinist. Therefore help was needed and the decision to hire someone was made. Jay Adeff, a member of NPS Physics Department, was hired to help with the physical design of the case, slit holder, and scanning mirror assembly. Jay took my dimensions and sketches and turned them into professional blue prints. Jay's assistance was invaluable, without which NUVIS would not have been completed.

With completed blue prints we needed a machinist. Glen Harrell, a member of the NPS Space Systems Academic Group, was hired to do the machining of NUVIS. Once the machining was completed the parts were anodized black. Figures 16 and 17 show the completed NUVIS. As can be seen, extensive weight relieving was used over the entire exterior of NUVIS.

The final dimensions of NUVIS were 62 cm by 39 cm by 37 cm by 28 cm by 26 cm by 19 cm high. The final mass was about 14 kg.

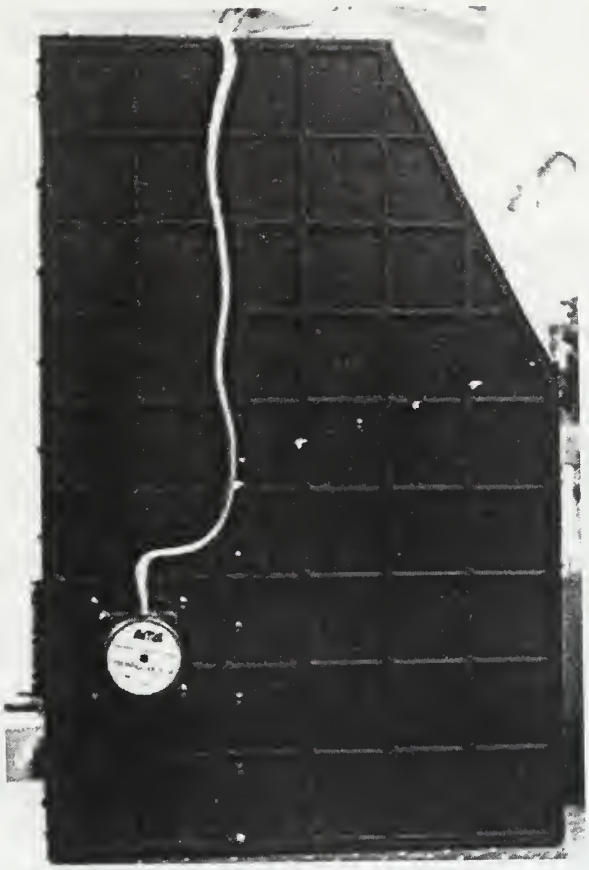


Figure 16: Top View of NUVIS.

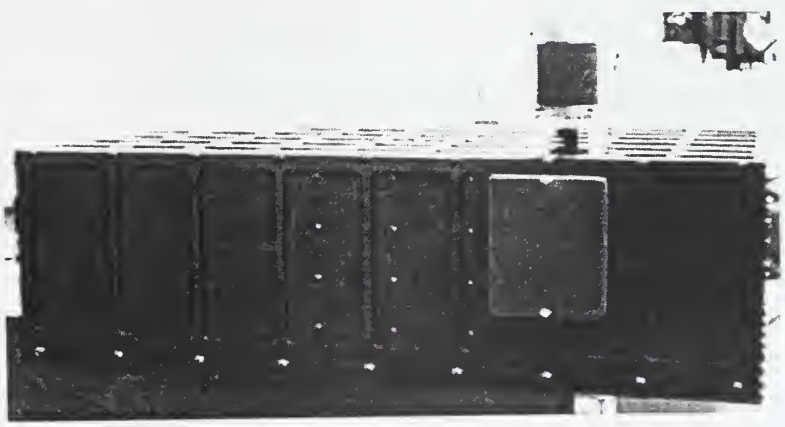


Figure 17: Side View of NUVIS.

III. ELECTRICAL-MECHANICAL COMPONENTS

A. STEPPING MOTOR

The instantaneous field of view produce a frame that included the vertical spacial dimension and the corresponding horizontal spectral components. In order to obtain a two-dimensional image with the corresponding spectral component, multiple frames, that were horizontally separated by one horizontal IFOV, were combined to produce a hypercube. Figure 18, shows the graphical representation of a hypercube.

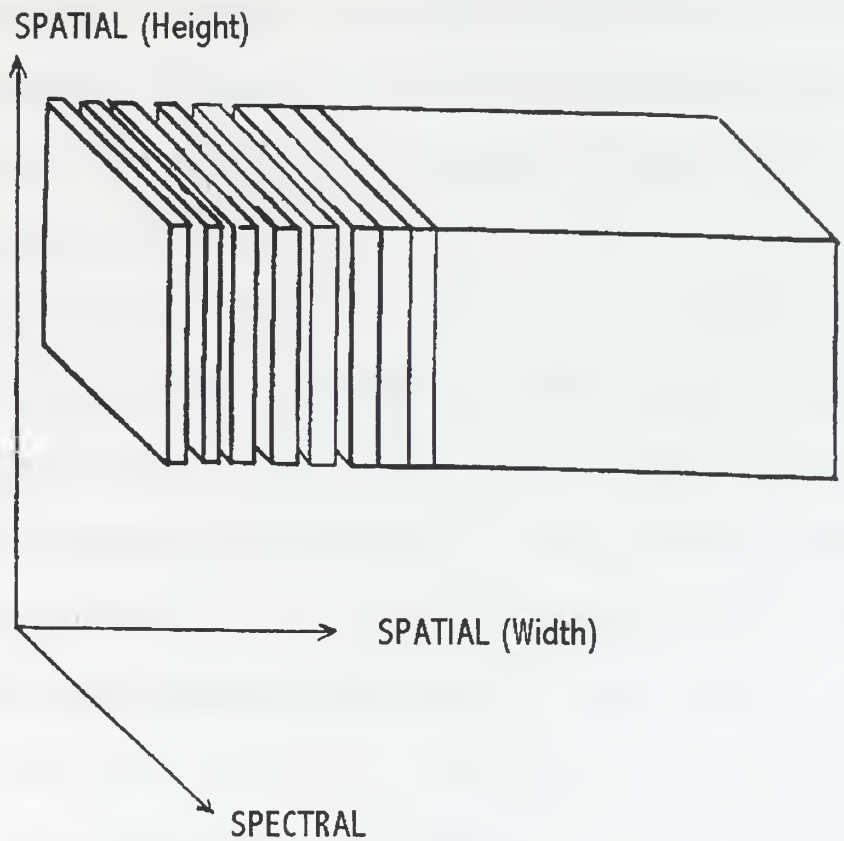


Figure 18: Graphical Representation of a Hypercube. After Johnson (1996).

The construction of a hypercube was accomplished by moving the scanning mirror half the horizontal IFOV. As discussed above, the horizontal IFOV was .59 mradians. This required that the mirror be moved approximately .3 mradians per frame. Converting the radians to steps, it was found that the scanning mirror must scan at about 21,000 steps per revolution. There were two options available to moving the scanning mirror: servomotor or a stepping motor.

After reviewing both types of motors and their respective controllers, I decided to use a stepping motor and micro stepping motor controller/driver. The stepping motor had several advantages over the servomotor: (1)stepping motors and micro stepping controller/drivers are more common than the servomotor and its controller, (2)servomotors and servomotor controllers with the required resolution and require torque were not available, and (3)servomotor controls were not as accurate as current micro stepping motor controllers.

The stepping motor, part number AM23-150-2, and the stepping motor controller/driver were purchased from Advanced Micro Systems (AMS).

The motor was a Frame Size 23, bipolar, 4 lead, two phase, series wound, hybrid stepping motor. It has a natural 1.8 degree per step which equates to 200 steps per revolution, with 150 oz-in holding torque and 3.3 oz/in-sec^2 rotor inertia. It operates at a maximum of 10 volts and draws a maximum current of 2.8 amps. It has a 6.4 mm diameter shaft with a length of 20.5 inch length.

The motor was chosen by first estimating torque needed to turn the scanning mirror 9 mradians per second. Nine mradians per second corresponds to 30 IFOV per second. This was driven by the camera's 30 fps maximum frame rate. The torque value was then tripled

to take into account the scanning mirror holder that had not yet been designed. Given this value I chose the a stepping motor that had a 15 percent higher torque value than needed.

NUVIS was initially designed to scan horizontally, so my calculations for the scanning mirror torque were for horizontal scanning, i.e., gravity was not a factor. Later, when given the opportunity to image the plumes of solid rocket motors at NPS Rocket Range, we needed to scan vertically. The motor torque was insufficient to step and hold the scanning mirror during microstepping at 25,600 steps per revolution. The stepping motor needed to be replaced with a higher torque motor and the scanning mirror and the scanning mirror holder needed to be balanced.

The motor was replaced with a AM23-210-3 motor, with an incremental encoder option, providing 210 oz-in torque. The new motor had a 40 percent torque increase and it added an incremental encoder.

Without the incremental encoder, step indexing was determined from the step pulses ordered by the stepping motor controller. With the incremental encoder, the step indexing came from the actual number of steps that the motor turned, thus creating a close loop control. This eliminated step slippage caused by the high torque load of the scanning mirror when scanning vertically.

B. STEPPING MOTOR CONTROLLER/DRIVER

The stepping motor controller/driver is shown Figure 19 and, like the stepping motor, was purchased from AMS. The model was a MAX-410 with incremental encoder option. It was a stand alone driver that interfaces to a PC or dumb terminal through a RS-422 to RS-232 proprietary adapter, part number SIN-8A. An RJ-45 connector with an 8-lead cable

connected the adapter to the controller. The controller/driver used standard 110V AC line power. The controller input and outputs include a five line connector for the stepping motor, a RJ-45 connector for connecting additional controllers in series, an incremental encoder input option, and five digital input/output lines or triggers.

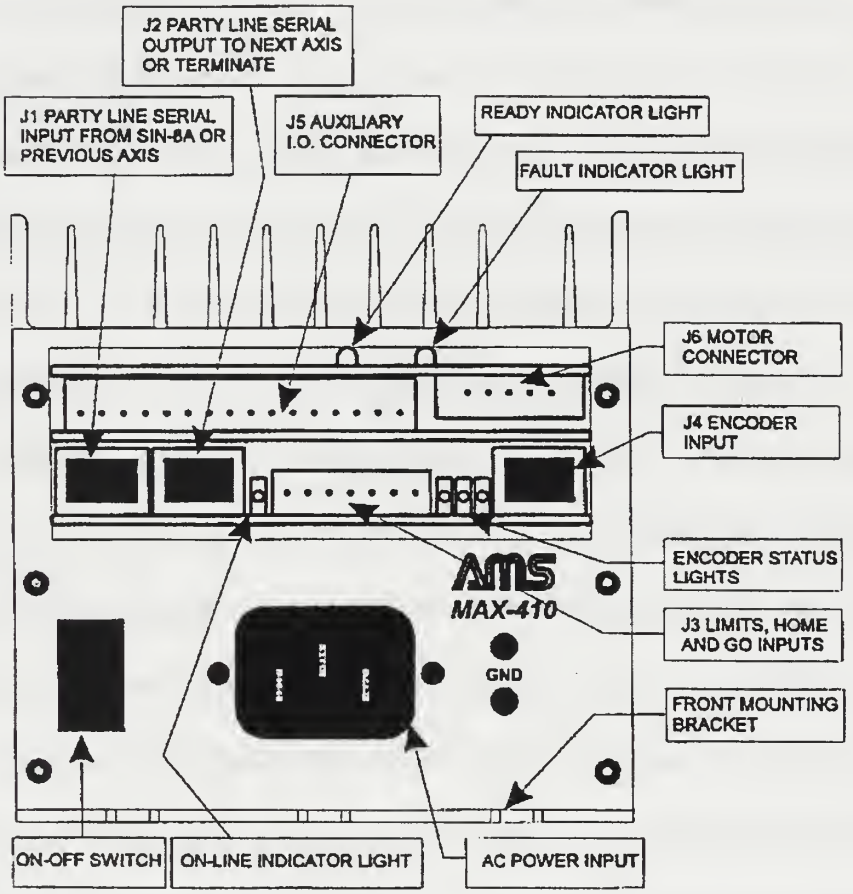


Figure 19: Max-410 Stepping Motor Controller/Driver. From Advanced Micro Systems (1996).

The MAX-410 controller used basic ASCII serial commands. The software included simulates a dumb terminal. During testing, a Wyse dumb terminal was used.

The controller had the capability of providing 40V at 4 amps. It can step at 200 - 51,200 steps per revolution in increments of multiples of two (200, 400, 800, etc.). As

discussed above, the controller provides internal indexing that was based on the number of steps sent or if an incremental encoder was used then on the actual number of steps detected by the encoder. The controller provides full control over acceleration, deceleration, ramp up speed, slow down speed, jog speed, running current usage and, holding current usage, just to mention a some of the controller's capabilities.

In our application the resolution was set to 25,600 steps per revolution, which corresponds to 0.25 mradians per step. After the completion of a command, one of the trigger outputs on the controller changes state. This output was used to trigger the National Instruments frame capture card, discussed below, to capture a image.

C. ABSOLUTE ENCODER

I needed some method of determining the scanning mirror position at power up. I determined that the easiest, most reliable, and most accurate way to do this was to use an absolute optical encoder. There are a large number of incremental encoder manufactures, but very few that manufacture absolute encoders. My search lead me to a company called US Digital Corp.

US Digital's absolute encoder, called the A2 , has a resolution from 2 to 65,536 steps per revolution, and allows an origin to be set using nonvolatile memory. The encoder was available in several variants, such as sleeve shafted and bearing shafted. The model I chose was the A2-S-K-250, a kit form of the encoder that had no shaft, but would except an external 6.4 mm shaft that was 15.2 - 20.3 mm long. The kit included the encoder and an optical encoder disk that had a 6.4 mm center hole with a set screw. Figure 20, shows a picture of the disassembled encoder.

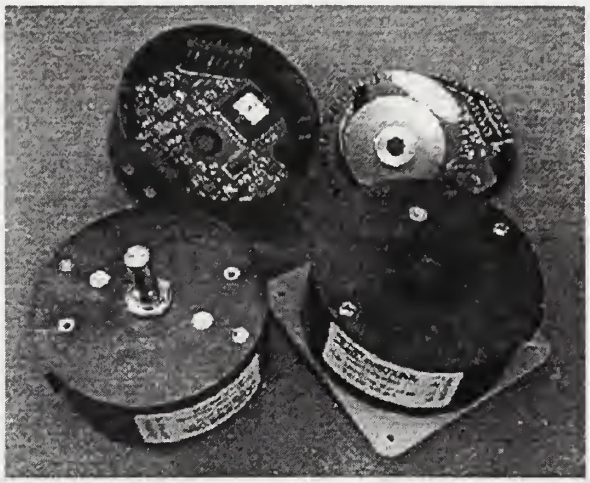


Figure 20: A2 Absolute Encoder. From US Digital.

The control and operation of the encoder was through a RJ-75, 6-pin connector that connected to a proprietary RS-232 adapter. The adapter also required a 9V external power source that was provided . (During operation of NUVIS, it was discovered that the adapter needed a power supply between 6 - 18 V. The significance of this will be explained below.)

A test platform was built using a 102 mm by 102 mm piece of aluminum angle iron. On one side a Frame Size 23 stepping motor mount was machined, while on the opposing side, holes were drilled and tapped for the encoder. The stepping motor and encoder were attached, with the encoder using the motor shaft. This test stand allowed me to test and experiment with both the encoder and the stepping motor and its controller/driver. The test procedure I used was simple, but it would allow me to fully test both components.

I first set both the encoder and the stepping motor origins so that they coincided. Then I wrote a short program, using the stepping motor controller/driver built-in language, that would generate a random number of steps in either the clockwise or counter clockwise directions. I then allowed this program to run for over 24 hours. At the end of the test, I

compared the stepping motor index with the encoder index. After nearly one billion steps, the encoder and stepping motor index still coincided exactly. I felt that the repeatability of the two components was more than adequate.

IV. ELECTRICAL COMPONENTS

A. IMAGE INTENSIFIED CCD CAMERA

1. CCD Camera

The search for a CCD camera was exhaustive. After nearly two months of research into cameras, manufactured by over two dozen companies, I decided on the Pulnix TM-745e. This decision was somewhat arbitrary. Most manufactures had cameras that were indistinguishable from one another. The requirements that I finally decided on were a black & white camera with a 2/3 in. CCD that had more than 640 by 480 pixels, remotely controllable electronic shutter speed, on chip frame integration, at least 8 bit grayscale, and standard RS-170 video interface. These requirements need some explanation.

NUVIS was designed with a total horizontal FOV of 20 degrees. This means that the scanning mirror must scan through an arc of 10 degrees. Using the 25,600 steps per revolution a 10 degree arc corresponds to 711 frames for the total horizontal FOV. This meant that a high speed camera was essential in order to take data in a reasonable amount of time. Even though speed was critical, I felt that low light conditions may require frame integration. I wanted a camera that used a digital interface to transfer the video data frame by frame. This type of camera proved inadequate because most single-frame, digital-interface, cameras could not capture and transfer at more than a few fps. Cameras that did have a high speed digital interface required a special computer interface that, when combined with the camera made the system cost unresonable. The solution was to use a video camera, with a standard RS-170 interface, that would provide up to 30 fps.

A video camera had several advantages and disadvantages. The advantages were that video cameras were relatively inexpensive and were readily available from dozens of manufactures. Also, the frame capture computer cards for RS-170 interface video cameras were about one-third the price of the digital interface card needed for digital cameras. The disadvantages are that we are limited to 8 bits of grayscale, 640 by 480 lines of resolution, and a more limited control of the image exposure.

Having decided to use a video camera, I next focused my attention to the types available. Larry Giligan (private conversations,1997) of Electro Optic Services, the company which coupled the camera and intensifier and installed the intensifier power supply, recommended that I purchase a camera with a 2/3 inch CCD instead of the more common ½ inch CCD. He explained that the optical coupling of a 25 mm image intensifier to the 2/3 inch CCD was easier and more efficient than that of the smaller ½ inch CCD. The 2/3 inch CCD requirement narrowed my search to only a handful of manufactures.

The requirement for on-chip frame integration was the final deciding factor. Only one manufacture had a camera that met all these requirements--Pulnix Inc.

The Pulnix TM-745e, as mention above, has a 768 by 494 2/3 inch CCD that measures 10 mm by 9.3 mm and has a pixel size of 11µm by 13 µm. As shown in Figure 21, the camera had three connectors, a BNC, and proprietary 6-pin and 12-pin.

The 6-pin connector and its pinouts are shown in Figure 22. It was used to control the shutter speed through a 3-bit TTL compatible signal. The shutter speeds available are listed in Table 1. A digital I/O card, discussed below, was used to control both the camera shutter speed and the frame integration.

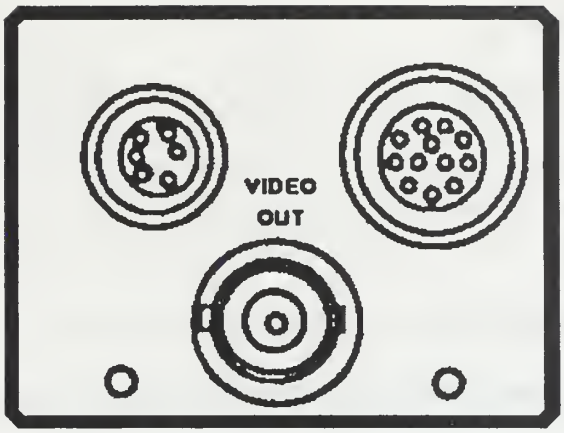
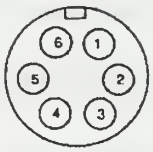


Figure 21: Camera Connectors. From Pulnix (1996).



6-PIN Connector

1. D2	4. +12V DC
2. GND	5. D0
3. Video	6. D1

NOTE: Do, D1, D2 are shutter control inputs.

Figure 22: Camera P6, Six Pin Connector. From Pulnix (1996).

Speed	1/60	1/125	1/250	1/500	1/1000	1/2000	1/4000	1/10000
D0	L	H	L	H	L	H	L	H
D1	L	L	H	H	L	L	H	H
D2	L	L	L	L	H	H	H	H

Table 1. Camera Shutter Speed Control. From Pulnix Corp. (1996).

The 12-pin connector, shown in Figure 23, supplied the camera power, 12 V, as well as optional horizontal and vertical sync, frame integration control bit, and option video output. The video was carried via a RG-59, 75 ohm, coaxial cable connected to the RS-170 interface using a standard BNC connectors. Frame integration was accomplished by providing a logic level low to the integration pin. As long as the logic level was low, charge will continue to accumulate on the CCD.

12-PIN Connector

1. GND	7. V_D In
2. +12V DC	8. GND
3. GND	9. H_D In
4. Video Out	10. N/C
5. GND	11. Integration Control
6. Vinit	12. GND

Figure 23: Camera P12, 12 Pin Connector. From Pulnix (1996).

2. Image Intensifier

As mentioned above the CCD camera and the image intensifier were sent to Electro-Optical Service, Inc to be optically coupled and have a image intensifier power supply added. Electro-Optical Services returned the components as a single unit, image intensified camera.

The image intensifier power supply required 12 +/- .5 V. It also has an input for a 0 - 10 V reference to control the MCP voltage (0V corresponding to 0 V across the MCP, and 10 V corresponding to 1600 V across the MCP). Electrical connections were made through a DB-9 connector on the image intensifier power supply housing. The pin out is listed in Table 2. The reference voltage was provided by a Digital-to-Analog-Converter (DAC) computer card, discussed below.

DB-9 Pin	Function	Description
1	Ground	Ground
2	Power Input	+ 12 (+/- 0.5) V
6	Gain Control	0 to 10 V

Table 2. Image Intensifier DB-9 Pin Out. From Electro-Optical Services (1997).

B. FRAME CAPTURE CARD

In order to get the data from the analog RS-170 video format to a digital format that could be analyzed, a video frame capture card was required. B&W frame capture cards vary

greatly in both quality and cost, but high quality does not always mean a high cost.

Frame capture cards are relatively new, and as a result the better cards have not had the time to break out from the poorer quality cards. An import factor that I discovered when considering a frame capture card was that the quality of the controlling software and documentation are just as important as the quality of the hardware itself. The frame capture cards can be classified into two basic classes, those with built-in VGA display drivers and those without.

At first, the cards with the built-in VGA display drivers appeared to be the better choice. These cards share the video memory. Part of the video memory was used to capture and hold the image while the remainder was used by the computer display. This arrangement allows for very fast and efficient image display. While one image was captured the previously captured image was displayed, all without the need to move the images into system memory. This arrangement prevented loading down the PCI bus and therefore freed the computer to work on other tasks. The only time that data would be required to be transferred over the computer's PCI bus was during actual recording. With little experience, I decided to purchase a frame capture card with built in VGA driver from the same distributor, JKN Electronics, Inc., where we purchased the camera. This card had the added benefit that it had an interface that allowed direct control of the Pulnix TM-745e camera shutter speed and frame integration.

I installed the card we purchased in a computer, loaded the drivers and example program and connected the camera. Everything worked great, from the display to the camera control. Believing one problem had been solved my attention turned to other components.

About a month latter it was time to begin programming. This was when I discovered that documentation and software were as important if not more important than the hardware.

The documentation that came with the card was sparse and what there was written for an expert Visual C++ programmer. After two months of encountering one problem after another with this card, I ran up against one final obstacle. I had discovered that there was a trade-off between having the VGA driver on the frame capture card and having a frame capture card alone.

The problem was that in this type of card the captured image was held in video memory. To save the image to a disk, the image must first be copied from video memory to system memory byte by byte. C++ did not have the lower functions needed to quickly and efficiently transfer the images from the video memory to system memory. Once in system memory, C++ provided ample speed and ease to do what ever I wanted. The result was that when transferring images from video memory to system memory the frame rate dropped from 30 fps to less than 5 fps. This problem coupled with the complexities of Visual C++ lead me to look for another solution.

What I found was a new card from National Instruments, IMAQ PCI-1408, that supported numerous programming languages including Visual C++ and Visual Basic. The card was capable of a full 30 fps with 8 bits of grayscale and it was fully documented with excellent manuals and numerous example programs.

The IMAQ was a PCI BusMaster card that has two external connectors. BusMaster means that the card can take control of the PCI bus and conduct data transfers without processor interaction. One of the cards connectors was a standard BNC for RS-170 video.

The other connector was a DB-15 that had three digital I/O lines that could be used as either digital I/O or triggers, horizontal and vertical sync input, and input for four video sources. The only draw back to this card was the difficulties capturing integrated frames.

The IMAQ only had horizontal and vertical inputs, no outputs. With no output, it had to take its timing from the video signal. Therefore without any way to control the sync timing signals there was no easy or efficient way to time the capture of an integrated frame. There were two ways around this. One was to use an external controllable sync timing source to provide the timing signals to both the camera and the IMAQ card. The other fix was to use software integration. Since the sensitivity of the camera and intensifier were found to be better than expected, integration should not be needed. But, if integration is required the software solution will be employed.

With the issue of the frame capture card settled, I still had not mastered the visual side of Visual C++. My solution was to abandon Visual C++ altogether and use Visual Basic. This created a trade-off. Visual Basic made the visual design much easier, but the language was not as powerful as C++, especially its inability to handle pointers. Despite the trade-off, Visual Basic proved to be a good choice.

C. DIGITAL I/O CARD

In order to control the camera shutter speed and frame integration, I needed four TTL logic bits. I was familiar with Keithley Metrabyte cards, but after having such success with the National Instruments IMAQ card, I decided to look at what National Instruments offered. What I found was the PC-AO-2DC, a combined 16 bit digital I/O card and 2 channel DAC. The driver software and examples included were excellent, and unlike Keithley Metrabyte,

National Instrument cards are Plug-n-Play, making installation a breeze. The only drawback was that the National Instrument card did not provide access to the computer's 12 V bus. For this reason, and this reason alone I decided to use a card from Keithley Metrabyte.

It might seen strange that I would make my decision solely on the voltage that a card can supply, but this was easily explained. NUVIS, was a field instrument, therefore the fewer the external support components required the better. If I had used the National Instruments card I would also need an additional 12 V power supply for the camera, image intensifier, and the absolute encoder.

The Keithley Metrabyte card that I chose for the digital I/O was the PIO-24. It was an ISA/EISA card that was functionally equivalent to the 8255 PPI, providing 24 bits of digital I/O in three ports (A, B, and C). Its I/O lines provided a high of 20 mA per bit compared to the normal 12 mA per bit.

The installation required that you find an available bus address and then set the dip switches on the card to that address. The PIO-24 address was Hex 300. This process was repeated for the IRQ setting, if the card's interrupts are to be used. I had no reason, nor saw future reason, to use interrupts, therefore the interrupt setting was disabled. After setting up the card, it was inserted into a free ISA or EISA slot. That was all that was required for setup. The card's control turned out to be more difficult.

In C and C++ the I/O commands allow direct access to the computer bus, but Visual Basic has no such commands. I had two options, attempt to find I/O commands from Keithley Metrabyte or write drivers myself. I chose the latter because it gave me full control, and allow me to write commands that mirrored C and C++.

Since the PIO-24 was functionally equivalent to the 8255 PPI, it requires a setup command to configure the three I/O ports. Table 3, shows the configuration bytes for the PIO-24. The base address of the PIO-24 was the address of port A. Port's B and C addresses follow sequentially. The configuration byte address was base plus three.

PA	PB	PC Upper	PC Lower	D7	D6	D5	D4	D3	D2	D1	D0
OUT	OUT	OUT	OUT	X	X	X	L	L	X	L	L
OUT	OUT	OUT	IN	X	X	X	L	L	X	L	H
OUT	OUT	IN	OUT	X	X	X	L	H	X	L	L
OUT	OUT	IN	IN	X	X	X	L	H	X	L	H
OUT	IN	OUT	OUT	X	X	X	L	L	X	H	L
OUT	IN	OUT	IN	X	X	X	L	L	X	H	H
OUT	IN	IN	OUT	X	X	X	L	H	X	H	L
OUT	IN	IN	IN	X	X	X	L	H	X	H	H
IN	OUT	OUT	OUT	X	X	X	H	L	X	L	L
IN	OUT	OUT	IN	X	X	X	H	L	X	L	H
IN	OUT	IN	OUT	X	X	X	H	H	X	L	L
IN	OUT	IN	IN	X	X	X	H	H	X	L	H
IN	IN	OUT	OUT	X	X	X	H	L	X	H	L
IN	IN	OUT	IN	X	X	X	H	L	X	H	H
IN	IN	IN	OUT	X	X	X	H	H	X	H	L
IN	IN	IN	IN	X	X	X	H	H	X	H	H

Table 3. PIO-24 Configuration Byte. From Keithley Metrabyte (1991).

The PIO-24 has a DB-37 connector with the pin out shown in Figure 24. Connection and use of this card was straight forward and no surprises were encountered.

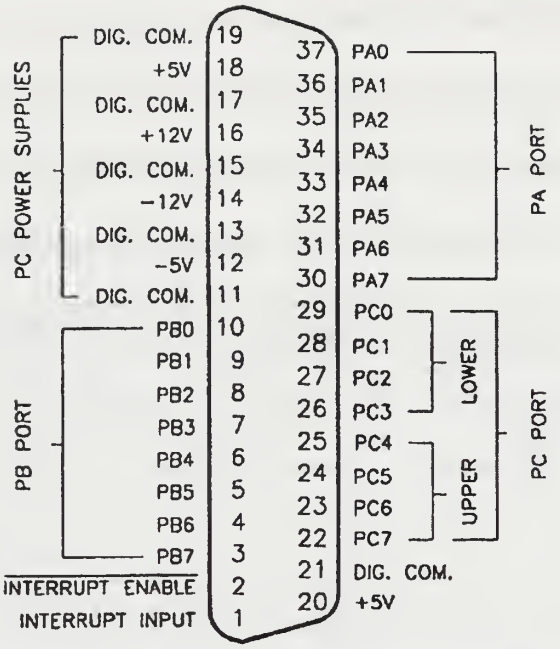


Figure 24: PIO-24 Connector. From Keithley Metrabyte (1991).

D. DIGITAL-TO-ANALOG-CONVERTER

In order to remotely control the image intensifier gain, and in conjunction with the shutter control, control the CCD camera exposure, a DAC was needed.

The first choice for this card was the National Instrument card discussed above. But with the selection of the PIO-24 as the digital I/O card, the addition of the National Instrument PC-AO-2DC card with 16 I/O lines and two 12 bit DAC taxed the available addresses on the computer bus. As a result, I again chose a Keithley Metrabyte card, the DAC-02.

The DAC-02 was an ISA/EISA card with the same type of installation as described above for the PIO-24, except there was no option for an IRQ setting. The base address for the DAC-02 card was Hex 390. This address corresponds to the lower byte of channel zero.

The DAC-02 provided a two-channel 12-bit DAC with output ranges of 0 - 10 V, 0 - 5 V, +/- 5 V, +/- 10 V, and a 4 - 20 mA current loop. The output was selected by jumpering the reference voltages on the cards DB-25 connector, as shown in Figure 25. Table 4 below shows the jumper pins for the desired output ranges.

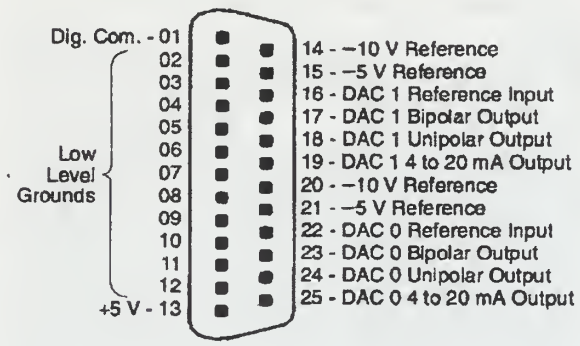


Figure 25: DAC-02 Connector. From Keithley Metrabyte (1994).

Range	DAC #	Jumper Pins	Output Pin
0 to 5 V	0	21 to 22	24
	1	15 to 16	18
0 to 10 V	0	20 to 22	24
	1	14 to 16	18
-5 to +5 V	0	21 to 22	23
	1	15 to 16	17
-10 to 10 V	0	20 to 22	23
	1	14 to 16	17

Table 4. DAC-02 Output Configuration. From Keithley Metrabyte (1994).

Control of the DAC-02 used the same functions as the PIO-24. In this case, each DAC channel used two addresses, a lower and upper address. The four most significant bits

(MSB) on lower address were the four least significant bits (LSB) in the 12 bit word. The eight MSBs of the 12 bit word come from the second address. The output of the channel does not change until the card receive two bytes, one on the lower address and one on the upper address of the corresponding channel. All four addresses for the two channels were sequential.

E. COMPUTER

The final electronic component was the control computer. The control computer should have been easy, but as with everything else involved with this instrument, it was not. What made it so difficult was the fact that the computer needed to be portable, rugged, and capable of accepting the PCI IMAQ-1408 card, the PIO-24 ISA/EISA card, and the DAC-02 ISA/EISA card. The need for I/O cards made using a laptop impossible but a standard computer was neither portable nor rugged. The only alternative was a portable style computer case that was common in the early 1980's. It turn out to be more difficult finding a portable computer than I had expected. I finally found a case, shown in Figure 26, distributed by a company called Case Depot.

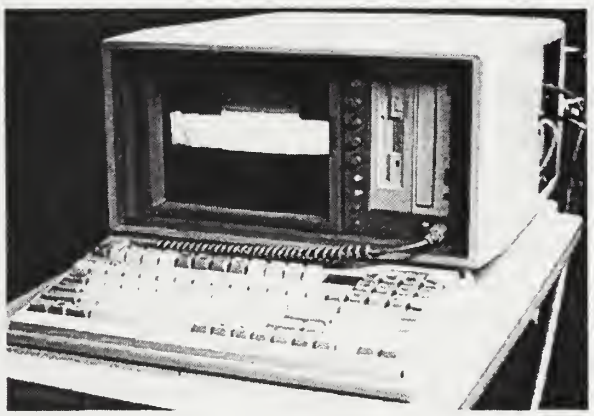


Figure 26: NUVIS Control Computer.

The case included a 10 inch Sony VGA monitor and keyboard that folded up covering the face of the computer. The case had two 5.25 inch drive bays, six expansion slots and a 230 Watt power supply. Along with the computer, I purchased a hardened shipping container, shown in Figure 27.

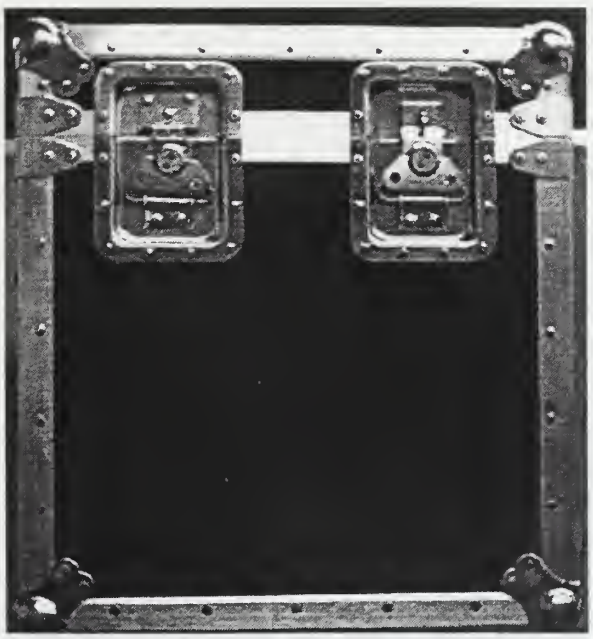


Figure 27: Computer Cargo Case.

I chose to install an ASUS P55T2P4 motherboard with a Pentium P5-200 processor in the portable case. This motherboard had an AT footprint that would fit in the case and it was the best rated Pentium motherboard on the market. The motherboard would accept up to 512 Mbyte of RAM, but due to the physical limitations of the case, only 128 Mbyte would fit. The hard drive was a 9.1 Gbyte Ultra-Wide SCSI Quantum drive. The SCSI controller was a Adaptec 2940-UW. The VGA card was an ATI Graphics Xpression+ with 2 Mbyte video RAM. A standard 3.5 inch drive and a 20x Toshiba CD-ROM were installed in the drive bays. The computer was assembled and tested in about week.

Figure 28 shows the connectors for the control computer. Starting in the upper left corner the connectors are the OS/2 mouse port and the parallel printer port. Starting in the lower left are com1 (DB-9) and com2 (DB-25), both RS-232 serial ports. The next connector was the VGA connector, seen with the monitor cable attached. Following this was a mini-68, SCSI-3 connector of the Adaptec 2940-UW. The IMAQ card connectors were next followed by the PIO-24 and DAC-02 connectors. As can be seen, there were no available expansion bays left. This has caused some inconvenience by preventing the installation of a LAN card, which would have been used for data transfers.

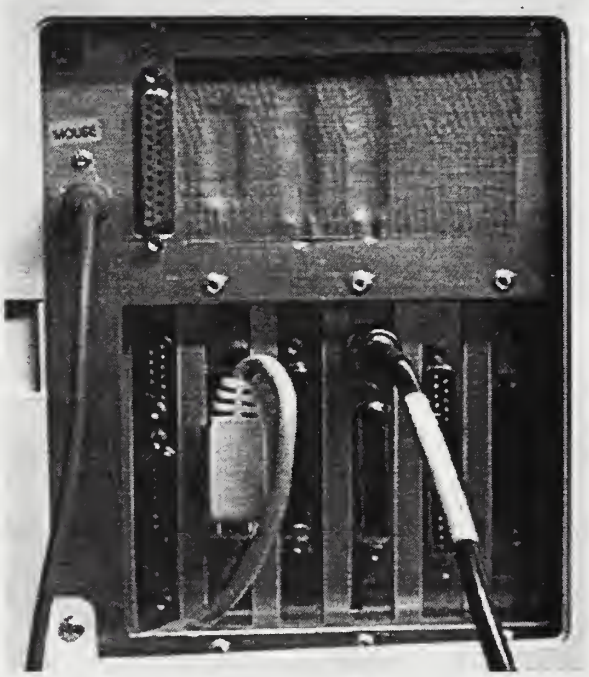


Figure 28: Computer Connectors.

Since each hypercube can take up to 109 Mbytes, data storage was critical. This was the reason for the 9.1 Gbyte hard drive, the largest available drive that would fit in the computer. But even with this large drive, there would not be enough storage for extended field use. The solution was to use an external storage device.

I envisioned that for field use, data would be taken during the day and stored on the hard drive. During the night the data would then be transferred to an external storage device. After review of the available external SCSI tape and disk drives I purchased an 8 Gbyte HP SureStore DAT SCSI tape drive and an Iomega 1 Gbyte SCSI JAZ drive. The tape drive would be used to store all the data, while the JAZ would be used to transport a limited amount of data from computer to computer.

No problems were encountered with the computer or its components during assembly, testing or extended operation.

F. CABLES AND WIRING HARNESS

1. Cable

The external cabling for NUVIS was somewhat involved. At the computer there were two RS-232 serial ports (DB-9 and DB-25), the IMAQ BNC connector, the PIO-24 (DB-37) connector, and the DAC-02 (DB-25) connector, shown in Figure 29. At NUVIS there was a BNC and three (DB-25) connectors. These are designated NUVIS, NUVIS 1, NUVIS 2, and NUVIS 2.

Figure 29: NUVIS Connectors.

The computer RS-232 serial ports, com1 and com2, are connected to the MAX-410 interface adapter and the A2 encoder interface adapter, mounted on NUVIS (NUVIS 2).

As discussed above, the MAX-410 connector was connected directly to the computer com2, (DB-25) serial port. From this adapter an RJ-45 connector with 8-lead cable connects the adapter to the MAX-410.

The other communications port, com1, serial port (DB-9), was connected to NUVIS via a DB-9 (computer) to DB-25 (NUVIS 2) cable. The encoder adapter resided inside NUVIS, on the other side of the DB-25.

The heart of the external cabling was the cable that connected the PIO-24 and DAC-02 to the first (DB-25) connector on NUVIS, called NUVIS 1. Table 5 shows the connections for this cable pin-for-pin. A third DB-25 connector, NUVIS 3, provides external access to digital I/O and voltage sources.

Notice that Port C Lower was used for controlling the camera speed and integration control. Port C Upper and Port A are only connected for future use. Note the digital and analog low level grounds that were tied to together. This was because the PIO-24 12 V supply was referenced to digital ground, therefore the 0-10 V signal from the DAC-02 also had to be referenced to the digital ground.

NUVIS	PIO-24	DAC-02	USE	NUVIS	PIO-24	DAC-02	USE
1	37	X	SPARE	14	24	X	SPARE
2	36	X	SPARE	15	23	X	SPARE
3	35	X	SPARE	16	22	X	SPARE
4	34	X	SPARE	17	21	X	GND
5	33	X	SPARE	18	20	X	+5V
6	32	X	SPARE	19	16	X	+12V
7	31	X	SPARE	20	14	X	-12V
8	30	X	SPARE	21	2	X	INT ENB
9	29	X	Shutter	22	1	X	INT
10	28	X	Shutter	23	X	24	Gain
11	27	X	Shutter	24	X	2,1	GND
12	26	X	Integrate	25	X	1,2	GND
13	25	X	SPARE				

Table 5. External Wiring Diagram.

2. Wiring Harness

The internal wiring harness pin-out is shown in Table 6. Spare indicates an unused pin. NUVIS 1 was the input DB-25. NUVIS 3 was an output DB-25 for future expansion. The +12V powers the image intensifier, camera, and the encoder adapter.

NUVIS 1	NUVIS 2	P6	P12	DB-9	SEI	Use
1	1	X	X	X	X	PA0, Spare
2	2	X	X	X	X	PA1, Spare
3	3	X	X	X	X	PA2, Spare
4	4	X	X	X	X	PA3, Spare
5	5	X	X	X	X	PA4, Spare
6	6	X	X	X	X	PA5, Spare
7	7	X	X	X	X	PA6, Spare
8	8	X	X	X	X	PA7, Spare
9	X	5	X	X	X	PC0, Shutter D0
10	X	6	X	X	X	PC1, Shutter D1
11	X	1	X	X	X	PC2, Shutter D2
12	X	X	11	X	X	PC3, Integrate
13	9	X	X	X	X	PC4, Spare
14	10	X	X	X	X	PC5, Spare
15	11	X	X	X	X	PC6, Spare
16	12	X	X	X	X	PC7, Spare
17	13	X	X	X	X	GND
18	14	X	X	X	X	+5V, Spare
19	15	X	2	2	1	+12V, Power
20	16	X	X	X	X	-12V, Spare
21	17	X	X	X	X	Interrupt Enable, Spare
22	18	X	X	X	X	Interrupt, Spare
23	19	X	X	6	X	0 - 10 V, Gain Control
24	20	2	1	1	1	GND, Ties analog and digital GND
25	21	X	X	X	X	GND, Ties analog and digital GND

Table 6. Internal Wiring Harness.

V. SOFTWARE

A. VISUAL C++

When I started the design of NUVIS, I fully expected to use Visual C++ for the software development. I was thoroughly familiar with ANSI standard C and with C++. I felt that with the aid of several good programing books and tutorials I would be able to learn what I needed to design the Graphical User Interface (GUI) that I envisioned for NUVIS control. As discussed above, this assumption was wrong. I did find several references, that did a good job of teaching the visual side of the programming, but these books all but ignored the integration of the actual code. After several months of very slow progress, I decided that there had to be a better way to get the job done. What I found to use instead of Visual C++ was Visual Basic (VB).

B. VISUAL BASIC

A new version of Visual Basic, Version 5.0, had just been released. It had a performance the was on par with earlier versions of C++, but the visual side of programming was much simpler.

Visual Basic is a powerful compiler based programing language that allows the development of Windows programs using a simpler development environment than Visual C++. Within a month of receiving the newly released version of VB, I had developed a windows program, shown in Figure 30, that allowed me to view live video, capture and save images, and capture and save sequences of images.

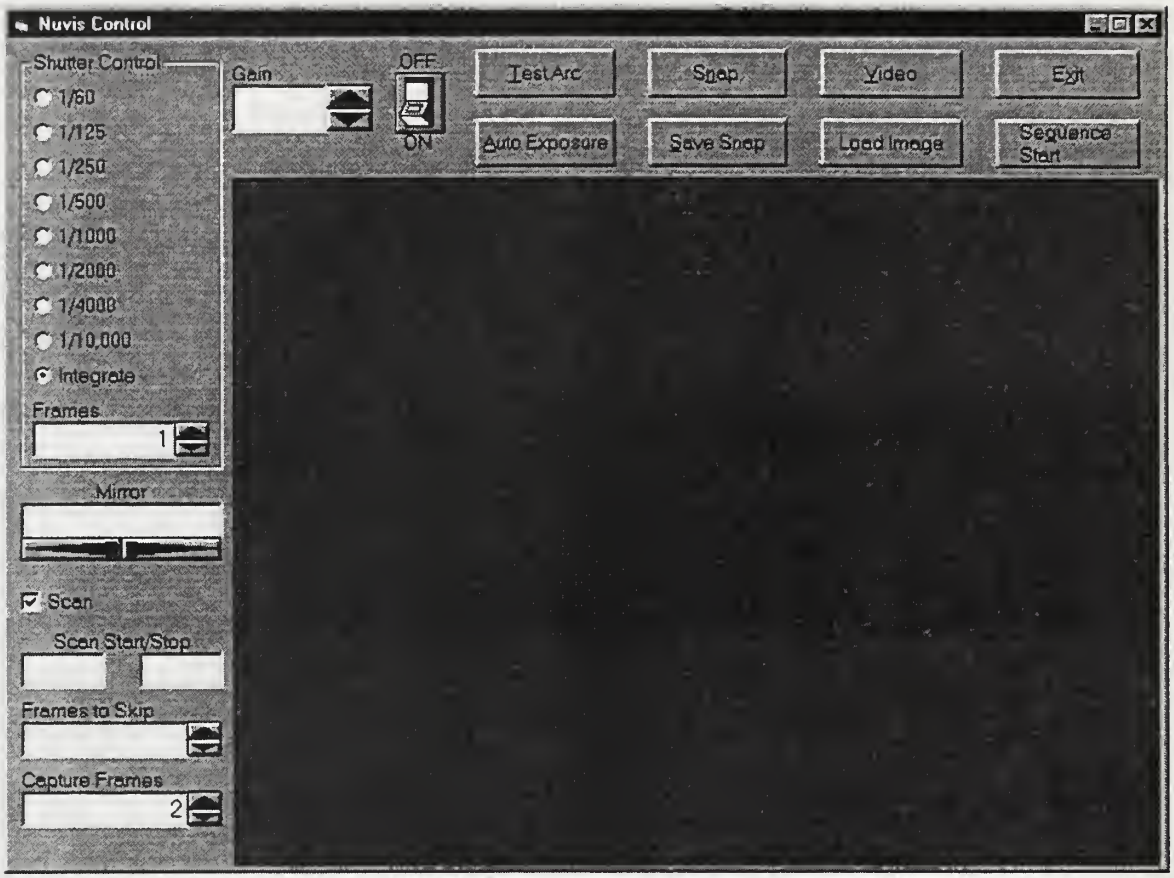


Figure 30: NUVIS Control Software Interface.

The program gives the user control of all NUVIS functions and view video from one dialog base window. As new components were added to NUVIS, I planned to add the corresponding controls and code to the program.

I first ran into trouble with the PIO-24 and DAC-02 cards. Visual Basic, although a powerful language, was not as powerful as Visual C++. The designers had decided that VB would not be allowed direct access to the computer bus. Keithley Metrabyte had included functions for VB that allow I/O, but these functions were very cumbersome to use, and their complexity was an over kill for this application. Without I/O controlling functions I would not be able to control the camera shutter speed and integration as well as the image

intensifier gain. I thought that by writing a Dynamic Link Library (DLL) in Visual C++, using the C++ I/O functions, I would be able to call the functions from VB. After several unsuccessful attempts I decided I needed some help. I began to search for a book that address I/O and VB. After a week of no luck, I broke down and called Microsoft. The technician went through a long explanation of why VB had no I/O functions and also why my attempts to write a DLL were unsuccessful. The bottom line, Microsoft did not want VB programers to have direct access to I/O and the VB calling procedure prevented the use of C++ I/O functions in a DLL. With this new information I decided to change my approach.

Years ago, while programming in FORTRAN 77, I needed I/O functions. The solution was to write the functions in Assembly language and then call them from FORTRAN. With the ability to write Assembly inline within C or C++ code, I felt that this was the simplest solutions. Therefore I started writing a DLL with the Assembly functions that were equivalent to the existing C and C++ I/O functions. After several unsuccessful attempts I looked for help. I found the help I needed in a book called Visual Basic 5.0 Programmer's Guide to the WIN32 API, Appleman (1997). This book included a CD-ROM that solved my problems, it contained a DLL (APIGID32.DLL) that among other things contained I/O functions written in Assembly language. This DLL gave VB the same I/O capability as C++.

C. IDL AND DYNAMIC LINK LIBRARIES

While I was working with Visual C++ and Visual Basic, an opportunity arose to use NUVIS to observe twelve solid rocket motor firings at NPS Rocket Range. With a deadline looming, we needed a software interface immediately. The VB program was not completed,

so an alternate option was to use IDL to take the data.

The idea of using IDL had been toyed with early on, but no frame capture card on the market had controlling functions that were callable from IDL. The problem was that IDL used a UNIX calling routine. In this UNIX calling form, two parameters are passed, one by value, the other by reference. The best way to explain how it works is to give an example. Suppose I want to pass two variables and four arrays to a function. The first parameter can be anything I want, but the second parameter was a pointer to an array of pointers. The second parameter points to the first element of an array whose elements are pointers that point to the data that was passed. In this example, the first two array elements point to the two variables while the next four array elements point to the first elements of the four arrays that were passed. This Unix type of argument passing was difficult and confusing to work with, and it was not support by the frame capture card manufactures. As a result, IDL was initially dismissed as the language for acquiring the data, although it will be the language used to analysis the data.

With a time crunch and me being the only person familiar with VB, IDL was again considered. Steve Finny and David D. Cleary of NPS Physics Department, were experienced with programming IDL, enough so that they felt they could build the GUI in several days. The problem was still getting IDL to call the National Instruments IMAQ functions for the frame capture card. The difference now was that I had gained experience making DLLs during my efforts to develop I/O functions for VB.

I first wrote a test function in a DLL that could be called by IDL. After this success, I began to incorporate the actual IMAQ functions, combining as many as possible in order

to minimize the required IDL calls. The result was a DLL that was callable from IDL, allowing IDL to access all the upper level functions of the IMAQ. While using the new DLL we found that IDL was not fast enough to keep up with the data rate flow.

IDL was receiving data at only about 4 fps, too slow to take data at a rocket firing that would only last about 8 seconds. The solution was to write functions in the DLL that allowed IDL to simply pass the number of frames to capture, and the pointer to the first element of the array that was to receive the data. After writing the new functions, IDL was able to record at 30 fps. The Appendix contains the printout of the NUVIS.DLL that was developed for IDL.

VI. SUMMARY AND RECOMMENDATIONS

When my thesis started, I had no idea of the difficulties that I would encounter. As the design and construction preceded, I would solve one problem only to be confronted by several more. As a result, the time it took to build NUVIS exceeded the estimates. But regardless of the difficulties, NUVIS was operational in time for twelve solid rocket motor firings that were held at the NPS Rocket Lab during the week of November 3, 1997.

When the time came to begin writing this thesis I realized that NUVIS was a living instrument. It would continue to undergo changes and modifications as experience is gained during its use. As a result I felt that this thesis would best serve as a sort of user's manual, containing detailed descriptions of the components, operation, and the pitfalls to avoid.

At the completion of this thesis, NUVIS still requires some work. Some of the work required was the result of experience gained during the rocket firings, other items just weren't completed due to the lack of time. I will continue to work on NUVIS for nearly two months after completion of this thesis, therefore the recommendations that are schedule for completion are annotated.

As discussed above, NUVIS was originally designed to scan horizontally. During the rocket firings, NUVIS was mounted vertically on a tripod. By changing the orientation, the effects of gravity on the scanning mirror could no longer be ignored. The result was that the stepping motor was unable to move or position the mirror accurately. The solution to this problem was to balance the scanning mirror and to replace the stepping motor with a more powerful motor that included an incremental encoder. As discussed above, a new stepping

motor with incremental encoder was installed but balancing the scanning mirror and scanning mirror holder has yet to be completed.

NUVIS construction was focused on looking at the UV region from 300 - 400 nm. By coincidence, the manufacture of the diffraction grating also makes gratings that are physical and optical replacements, providing different bandwidths. A new grating, with a 200 - 400 nm bandwidth will be ordered and in December 1997. In order to use the increased bandwidth of the new grating, a new filter will be needed. This filter most likely will have to be custom made, therefore simulations will be needed to provide a manufacture with the required transmission curve.

Even though the Visual Basic interface provides a good control for NUVIS, I recommend that a robust, multi-windowed application be either developed in house or be contracted out. The only way to take full advantage of the hardware capabilities is through the C or C++ language.

At a later date the current camera should be replaced with a 10 bit, or greater, grayscale camera. This will also require the replacement of the frame capture card. Therefore, a camera with a digital output and a frame capture card with a digital input should be pursued. The need for this was a result of experience. With only an 8-bit grayscale the dynamic range was narrow, making it very difficult to take images without saturating the CCD. A 10-bit, or greater, gray scale will help alleviate this problem.

APPENDIX . DYNAMIC LINK LIBRARY FOR IDL

The following file is the code file for NUVIS.DLL. NUVIS.DLL is a Dynamic Link Library that provides an interface rapping for IDL, allowing IDL to calling the controlling functions for National Instruments IMAQ frame capture card.

```
//-------------------------------------------------------------------------
//Filename:       Nuvis.cpp
//By:             LT Todd A. Hooks
//Date:           11/2/1997
//Purpose:        Provide an interface rapping for IDL and
//                National Instruments IMAQ Frame Capture Card
//Remarks:        Little to no error handling has been incorporated in
//                the following functions.  All functions operate.
//-------------------------------------------------------------------------

#include <windows.h>
#define _NIWIN

#include <iostream.h>
#include "niimaq.h"
#include "nitypes.h"


//Declare the DLL finctions prototypes.
long Test(int, int &);


////////////////////////////////////////////
//DLLEntryPoint():  The entry point of the DLL
////////////////////////////////////////////
BOOL WINAPI DllEntryPoint (HINSTANCE hDLL, DWORD dwReason,
                                                LPVOID Reserved)
{

switch (dwReason)
        {

        case DLL_PROCESS_ATTACH:
                {
```

```
                break;
                }

        case DLL_PROCESS_DETACH:
                {


                break;
                }

        }

return TRUE;
}


//Test Function to determining if IDL calling convention and this
//function are operating as expected.
LONG WINAPI Test(LONG numArguments, PULONG ppBuffer[])
{

        PULONG          pnumFrames = ppBuffer[0];
        ULONG           numFrames = *pnumFrames;
        PBYTE           pSkip = (PBYTE)ppBuffer[1];
        PBYTE           pBufferArray = (PBYTE)ppBuffer[2];
        UINT        i;

        for(i = 0; i < numFrames; i++)
        {
                *pSkip = 11;
                pSkip++;
        }

        for(i = 0; i < numFrames; i++)
        {
                *pBufferArray = 15;
                pBufferArray++;
        }

        return(0);
}
```

```
//--------------------------------------------------------------------------
//Function:             GrabOpen
//Parameters:  pointer to a pointer to sessionID
//                                      pointer to a pointer to interfaceID
//Return:               returns zero
//--------------------------------------------------------------------------
LONG WINAPI GrabOpen(LONG numArguments, PULONG ppBuffer[])
{
        ULONG               interfaceID;
        ULONG               sessionID;
        LONG        error;
        PULONG              pSessionID = ppBuffer[0];
        PULONG              pInterfaceID = ppBuffer[1];

        //Open an interface and a session
        error = imgInterfaceOpen("img0", &interfaceID);
        error = imgSessionOpen(interfaceID, &sessionID);

        //configure the session for a grab but do not start the acquisition yet
        error = imgGrabSetup(sessionID, FALSE);

        //start the acquisition
        error = imgSessionStartAcquisition(sessionID);

        *pSessionID = sessionID;
        *pInterfaceID = interfaceID;

        return(0);
}



//--------------------------------------------------------------------------
//Function:             GetGrab
//Parameters:  a pointer to a pointer to sessionID
//                                      a pointer to a pointer to a image buffer
//Return:               returns zero
//--------------------------------------------------------------------------
LONG WINAPI GrabImage(LONG numArguments, PULONG ppBuffer[])
{

        ULONG               sessionID;
        LONG        error;
```

```
        PULONG          pSessionID = ppBuffer[0];
        PBYTE           pBuffer = (PBYTE)ppBuffer[1];

        sessionID = *pSessionID;

        //grab a copy of the acquisition buffer into my own user buffer
        error = imgGrab(sessionID, &pBuffer, TRUE);

        return(0);
}


//--------------------------------------------------------------------------
//Function:         GrabClose
//Parameters:  a pointer to a pointer to sessionID
//                                  a pointer to a pointer to interfaceID
//Return:           returns zero
//--------------------------------------------------------------------------
LONG WINAPI GrabClose(LONG numArguments, PULONG ppBuffer[])
{

        ULONG           sessionID;
        ULONG           interfaceID;
        LONG        error;
        PULONG          pSessionID = ppBuffer[0];
        PULONG          pInterfaceID = ppBuffer[1];

        sessionID = *pSessionID;
        interfaceID = *pInterfaceID;

        //stop the grab acquisition
        error = imgSessionStopAcquisition(sessionID);

        //close this interface, free all resources
        error = imgClose(interfaceID, TRUE);

        return(0);
}
```

```
//-----------------------------------------------------------------------
//Function:             SnapOpen
//Parameters:  pointer to a pointer to sessionID
//                                      pointer to a pointer to interfaceID
//Return:               returns zero
//-----------------------------------------------------------------------
LONG WINAPI SnapOpen(LONG numArguments, PULONG ppBuffer[])
{
        ULONG               interfaceID;
        ULONG               sessionID;
        LONG        error;
        PULONG              pSessionID = ppBuffer[0];
        PULONG              pInterfaceID = ppBuffer[1];

        //Open an interface and a session
        error = imgInterfaceOpen("img0", &interfaceID);
        error = imgSessionOpen(interfaceID, &sessionID);

        *pSessionID = sessionID;
        *pInterfaceID = interfaceID;

        return(0);
}


//-----------------------------------------------------------------------
//Function:             SnapImage
//Parameters:  a pointer to a pointer to sessionID
//                                      a pointer to a pointer to a image buffer
//Return:               returns zero
//-----------------------------------------------------------------------
LONG WINAPI SnapImage(LONG numArguments, PULONG ppBuffer[])
{

        ULONG               sessionID;
        LONG        error;
        PULONG              pSessionID = ppBuffer[0];
        PBYTE               pBuffer = (PBYTE)ppBuffer[1];

        sessionID = *pSessionID;
```

```
        //grab a copy of the acquisition buffer into my own user buffer
        error = imgSnap(sessionID, &pBuffer);

        return(0);
}



//--------------------------------------------------------------------------
//Function:             SnapClose
//Parameters:  a pointer to a pointer to interfaceID
//Return:               returns zero
//--------------------------------------------------------------------------
LONG WINAPI SnapClose(LONG numArguments, PULONG ppBuffer[])
{

        LONG            error;
        PULONG                  pInterfaceID = ppBuffer[0];
        ULONG                   interfaceID;

        interfaceID = *pInterfaceID;

        //close this interface, free all resources
        error = imgClose(interfaceID, TRUE);

        return(0);
}



//--------------------------------------------------------------------------
//Function:             SequenceOpen
//Parameters:  a pointer to a pointer to sessionID
//                                      a pointer to a pointer to interfaceID
//Return:               returns zero
//--------------------------------------------------------------------------
LONG WINAPI SequenceOpen(LONG numArguments, PULONG ppBuffer[])
{
        LONG            error = 0;
        ULONG                   sessionID = 0;
        ULONG                   interfaceID = 0;
```

```
        PULONG          pSessionID = ppBuffer[0];
        PULONG          pInterfaceID = ppBuffer[1];

        //open an interface and a session
        error = imgInterfaceOpen("img0", &interfaceID);
        error = imgSessionOpen(interfaceID, &sessionID);

        *pSessionID = sessionID;
        *pInterfaceID = interfaceID;


        return(0);
}




//-------------------------------------------------------------------------
//Function:             SequenceImage
//Parameters:   a pointer to a pointer to sessionID
//                              a pointer to a pointer to numFrames
//                              a pointer to a pointer to skip
//                              a pointer to a pointer to buffer
//Return:               returns zero
//-------------------------------------------------------------------------
LONG WINAPI SequenceImage(LONG numArguments, PULONG ppBuffer[])
{
        CONST ULONG     MAX_FRAMES = 50;
        CONST ULONG     FRAMESIZE = 307200;

        LONG            error = 0;

        PULONG          pSessionID = ppBuffer[0];
        ULONG           sessionID = *pSessionID;

        PULONG          pnumFrames = ppBuffer[1];
        UONG            numFrames = *pnumFrames;

        PULONG          pSkip = ppBuffer[2];
        ULONG           skip = *pSkip;
        ULONG           skipArray[350] = {0};
```

```
ULONG               buffer[350] = {0};
PBYTE               pBuffer = (PBYTE)ppBuffer[3];
UINT        i, j;
UINT        numLoop, numRemainder;



if (numFrames > 350)
{
        return(1);
}

numLoop = numFrames/MAX_FRAMES;
numRemainder = numFrames%MAX_FRAMES;

for(i = 0; i < numFrames; i++)
{
        skipArray[i] = skip;
}

if (numLoop > 0)
{
        for(i = 0; i < numLoop; i++)
        {
                for(j = 0; j < MAX_FRAMES; j++)
                {
                        buffer[j] = (ULONG)pBuffer;
                        pBuffer += FRAMESIZE;
                }

                //configure the session for a sequence with numFrames, using the
                //buffer passed in pBuffer.
                error = imgSequenceSetup(sessionID, MAX_FRAMES,
(void**)buffer, skipArray, TRUE, FALSE);

        }
}

if (numRemainder > 0)
{
        for(i = 0; i < numRemainder; i++)
        {
                buffer[i] = (ULONG)pBuffer;
```

```
				pBuffer += FRAMESIZE;
			}

			//configure the session for a sequence with numFrames, using the
			//buffer passed in pBuffer.
			error  =  imgSequenceSetup(sessionID,  numRemainder,  (void**)buffer,
skipArray, TRUE, FALSE);
		}


		return(0);
}




//------------------------------------------------------------------------
//Function:			SequenceClose
//Parameters:	a pointer to a pointer to interfaceID
//Return:			returns zero
//------------------------------------------------------------------------
LONG WINAPI SequenceClose(LONG numArguments, PULONG ppBuffer[])
{

		LONG			error;
		PULONG				pInterfaceID = ppBuffer[0];
		ULONG				interfaceID = *pInterfaceID;

		//close this interface, free all resources
		error = imgClose(interfaceID, TRUE);

		return(0);
}


//End Nuvis.cpp
```

The following file, Nuvis.def, contains the functions defined in Nuvis.cpp

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; Nuvis.def
;
; The DEF file for the Nuvis.DLL DLL.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LIBRARY   nuvis

CODE      PRELOAD MOVEABLE DISCARDABLE
DATA      PRELOAD SINGLE

EXPORTS
  ; The names of the DLL functions
  Test
  GrabOpen
  GrabImage
  GrabClose
  SnapOpen
  SnapImage
  SnapClose
  SequenceOpen
  SequenceImage
```

```
SequenceClose

//End Nuvis.def
```

LIST OF REFERENCES

Advanced Micro Systems, *MAX-410 Technical Reference Guide*, (1996).

Appleman, Dan, *Visual Basic 5.0 Programmer's Guide to the WIN32 API*, Ziff-Davis Press, Emeryville, California (1997).

Clausing, Howard, P.A. Clausing, Inc., private conversations (1997).

Delft Electronische Producten (DEP), *Second Generation Image Intensifiers* (1994).

Electro-Optical Services, Inc., letter (1997).

Giligan, Larry, Electro-Optical Services, Inc., private conversations (1997).

Johnson, E. O., "Design, development, and testing of an ultraviolet hyperspectral imager", Master's Thesis, Naval Postgraduate School, Monterey California, (1996).

Keithley Metrabyte Corporation, *PIO-24 User's Guide*, (1991).

Keithley Metrabyte Corporation, *DAC-02 User's Guide*, (1994).

MacMannis, A. R., "The design of the Naval Postgraduate School's Ultraviolet Imaging Spectrometer (NUVIS)", Master's Thesis, Naval Postgraduate School, Monterey, California, (1997).

Pulnix, *Operations and Maintenance Manual of the TM-754/TM-765 High Resolution CCD Camera*, (1996).

Schott Glass Technologies, Inc., *Optical Glass Filters.*

Walden, B. S., "An analysis of middle ultraviolet dayglow spectra", Master's Thesis, Naval Postgraduate School, Monterey, California, (1991).

Ziemer & Associates, Inc., letter (1997).

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd. STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5000

3. Dr. David D. Cleary 3
   Physics Department, PH-CL
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. LT Todd A. Hooks 3
   C/O Michael A. Hooks
   R.D. #3 Box 398
   Kittanning, PA 16201

5. Dr. Donald Walters 3
   Physics Department, PH-WA
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Dr. William B. Meier II, Chairman PH 1
   Physics Department, PH-WB
   Naval Postgraduate School
   Monterey, CA 93943-5000